Restricted Boltzmann Machines for Collaborative Filtering

Ruslan Salakhutdinov

joint work with Andriy Mnih and Geoffrey Hinton

University of Toronto, Machine Learning Group

ICML, June 22 2007

Netflix Dataset

- The training data set consists of 100,480,507 ratings from 480,189 randomly-chosen, anonymous users on 17,770 movie titles.
- As part of the training data, Netflix also provides validation data, containing 1,408,395 ratings. A test set contains 2,817,131 user/movie pairs with the ratings withheld.
- Performance is assessed by submitting predicted ratings to Netflix who then post the root mean squared error (RMSE) on an unknown half of the test set.
- As a baseline, Netflix provided the score of its own system trained on the same data, which is 0.9514.
- Over 2,000 teams worldwide are attempting to achieve a 10% improvement over the Netflix's own score to win 1 million dollars.
- I will tell you how to get an almost 7% improvement.

Other Collaborative Filtering Methods

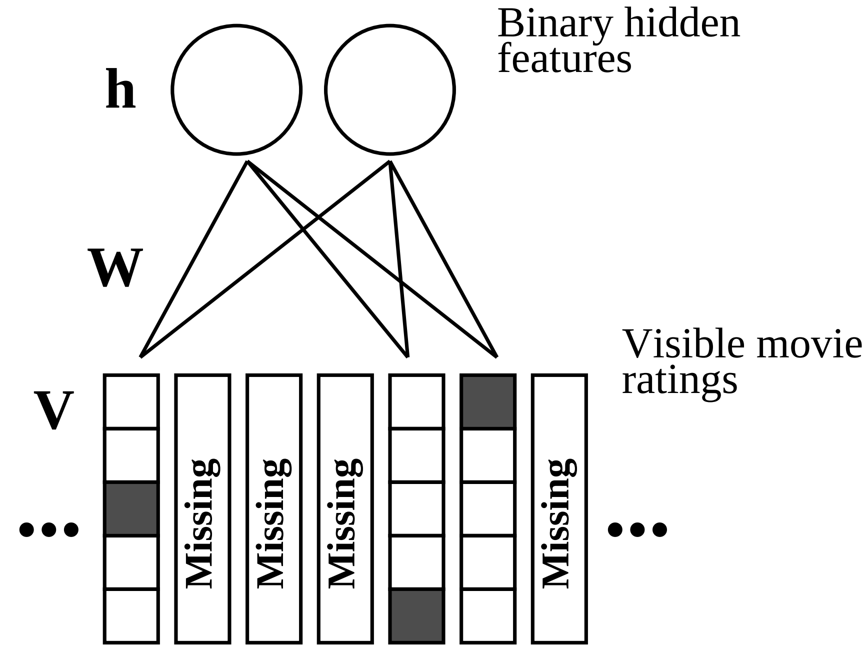
- Many researchers and practitioners have been attempting to carefully tune standard collaborative filtering methods, including:
  - nearest neighbor methods using Pearson's correlation
  - the user rating profile (URP) model
  - multinomial mixture models, and many others.

- None of these methods have proved to be particularly successful so far.

- Performance of these models on the Netflix dataset rarely surpasses a RMSE of 0.92-0.93.

- Very few collaborative filtering approaches scale well to large datasets.

Restricted Boltzmann Machines



- A restricted Boltzmann machine with binary hidden units and softmax visible units.

- For each user, the RBM only includes softmax units for the movies that user has rated.

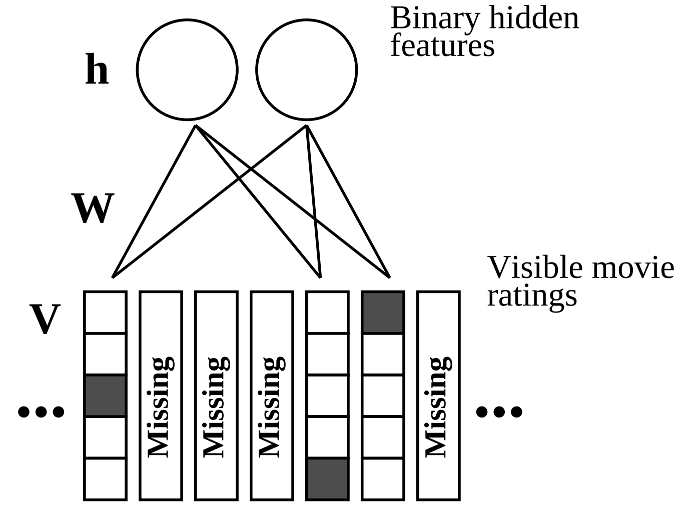
- Suppose a user rated m movies. Let $\mathbf{V}$ be a $K \times m$ observed binary indicator matrix with $v_i^k = 1$ if the user rated movie i as k and 0 otherwise.

- We also let $\mathbf{h}$ represent stochastic binary hidden features that have different values for different users.

Restricted Boltzmann Machines

- A joint configuration $(\mathbf{V}, \mathbf{h})$ has an energy:

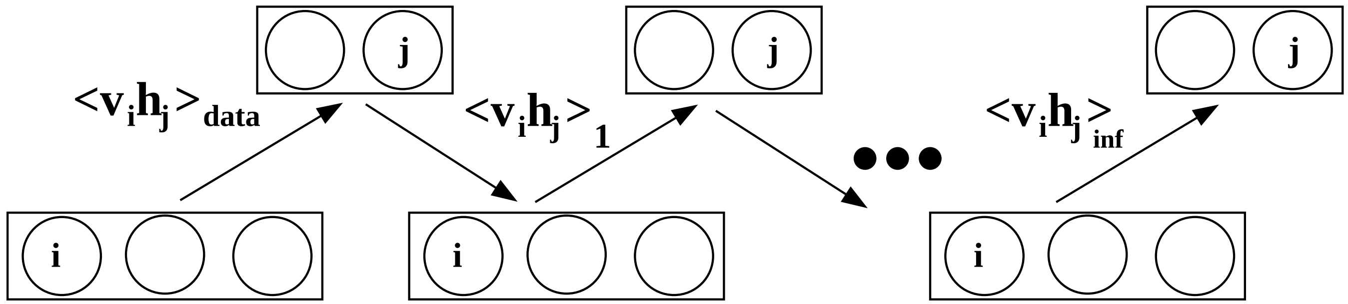
$$E(\mathbf{V}, \mathbf{h}) = -\sum_{i=1}^{m}\sum_{j=1}^{F}\sum_{k=1}^{K} W_{ij}^{k} h_j v_i^k + \\ -\sum_{i=1}^{m}\sum_{k=1}^{K} v_i^k b_i^k - \sum_{j=1}^{F} h_j b_j$$



- The probability that the model assigns to $\mathbf{V}$:

$$p(\mathbf{V}) = \sum_{\mathbf{h}\in\mathcal{H}} p(\mathbf{V}, \mathbf{h}) = \sum_{\mathbf{h}\in\mathcal{H}} \frac{\exp(-E(\mathbf{V}, \mathbf{h}))}{\sum_{\mathbf{V}',\mathbf{h}'} \exp(-E(\mathbf{V}', \mathbf{h}'))}$$

Inference and Learning



- Conditional distributions over hidden and visible units are given by:

$$p(v_i^k = 1|\mathbf{h}) = \frac{\exp\left(b_i^k + \sum_{j=1}^{F} h_j W_{ij}^k\right)}{\sum_{l=1}^{K} \exp\left(b_i^l + \sum_{j=1}^{F} h_j W_{ij}^l\right)}$$

$$p(h_j = 1|\mathbf{V}) = \sigma(b_j + \sum_{i=1}^{m} \sum_{k=1}^{K} v_i^k W_{ij}^k)$$

- Maximum Likelihood learning:

$$\Delta W_{ij}^k = \epsilon \frac{\partial \log p(\mathbf{V})}{\partial W_{ij}^k} = \epsilon\Big(< v_i^k h_j >_{data} - < v_i^k h_j >_{model} \Big)$$

- Contrastive Divergence (1-step) learning:

$$\Delta W_{ij}^k = \epsilon(< v_i^k h_j >_{data} - < v_i^k h_j >_1)$$

Making Predictions

- Given the observed ratings $\mathbf{V}$, we can predict a rating for a new query movie q in time linear in the number of hidden units:

$$p(v_q^k = 1|\mathbf{V}) \propto \sum_{h_1,\ldots,h_p} \exp(-E(v_q^k, \mathbf{V}, \mathbf{h}))$$

$$\propto \Gamma_q^k \prod_{j=1}^{F} \sum_{h_j \in \{0,1\}} \exp\Big(\sum_{i=1}^{m}\sum_{l=1}^{K} v_i^l h_j W_{ij}^l + v_q^k h_j W_{qj}^k + h_j b_j\Big)$$

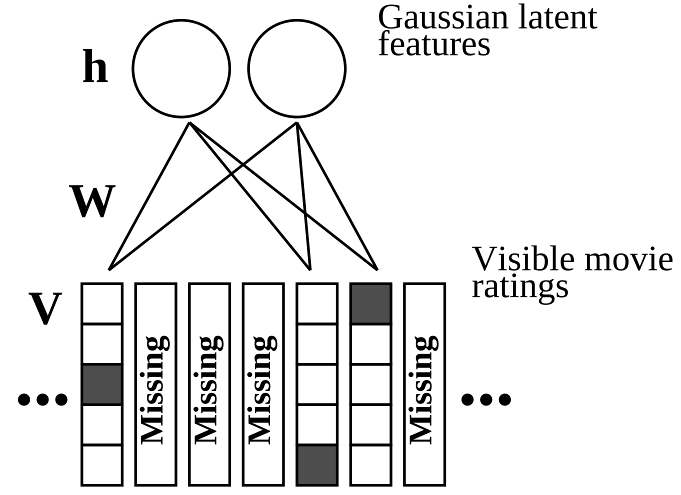
$$= \Gamma_q^k \prod_{j=1}^{F} \Bigg(1 + \exp\Big(\sum_{i=1}^{m}\sum_{l=1}^{K} v_i^l W_{ij}^l + v_q^k W_{qj}^k + b_j\Big)\Bigg)$$

where $\Gamma_q^k = \exp\left(v_q^k b_q^k\right)$.

- Once we obtain unnormalized scores, we perform normalization over K values to get probabilities $p(v_q = k|\mathbf{V})$ and take the expectation $E[v_q]$ as our prediction.

RBM's with Gaussian Hidden Units

- We can also model "hidden" user features **h** as Gaussian latent variables:



$$p(v_i^k = 1|\mathbf{h}) = \frac{\exp\left(b_i^k + \sum_{j=1}^F h_j W_{ij}^k\right)}{\sum_{l=1}^K \exp\left(b_i^l + \sum_{j=1}^F h_j W_{ij}^l\right)}$$

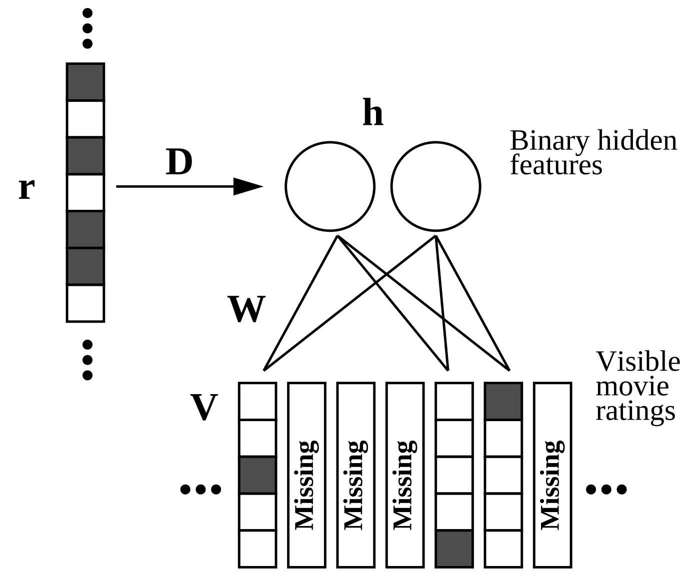
$$p(h_j = h|\mathbf{V}) = \frac{1}{\sqrt{2\pi}\sigma_j} \exp\left(- \frac{\left(h - b_j - \sigma_j \sum_{ik} v_i^k W_{ij}^k\right)^2}{2\sigma_j^2} \right)$$

where σ_j^2 is the variance of the hidden unit j.

- The energy takes form:

$$E(\mathbf{V}, \mathbf{h}) = -\sum_{ijk} W_{ij}^k \frac{h_j}{\sigma_j} v_i^k + \\ - \sum_{ik} v_i^k b_i^k + \sum_j \frac{(h_j - b_j)^2}{2\sigma_j^2}$$

Conditional RBM's



• We know in advance which user/movie pairs occur in the test set, so we have a third category: movies that were viewed but for which the rating is unknown.

• This is a valuable source of information about users who occur several times in the test set, especially if they only gave a small number of ratings in the training set.

• The binary vector **r**, indicating rated/unrated movies, affects binary states of the hidden units:

$$p(v_i^k = 1|\mathbf{h}) = \frac{\exp\left(b_i^k + \sum_{j=1}^F h_j W_{ij}^k\right)}{\sum_{l=1}^K \exp\left(b_i^l + \sum_{j=1}^F h_j W_{ij}^l\right)}$$

$$p(h_j = 1|\mathbf{V}, \mathbf{r}) = \sigma\left(b_j + \sum_{i=1}^m \sum_{k=1}^K v_i^k W_{ij}^k + \sum_{i=1}^M r_i D_{ij}\right)$$

Conditional Factored RBM's

- One disadvantage of the RBM models is that their current parameterization of $W \in R^{M \times K \times F}$ results in a large number of free parameters.

- In our current implementation, with $F = 100$ (the number of hidden units), $M = 17770$, and $K = 5$, we end up with about 9 million free parameters.

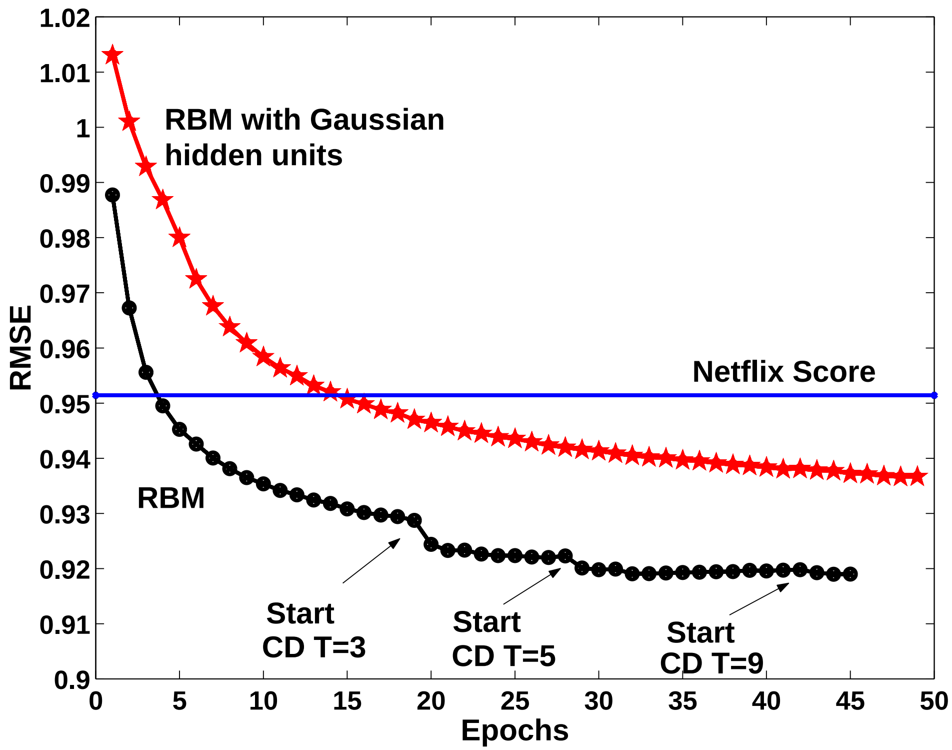
- We can factorize the parameter matrix W into a product of two lower-rank matrices A and B:

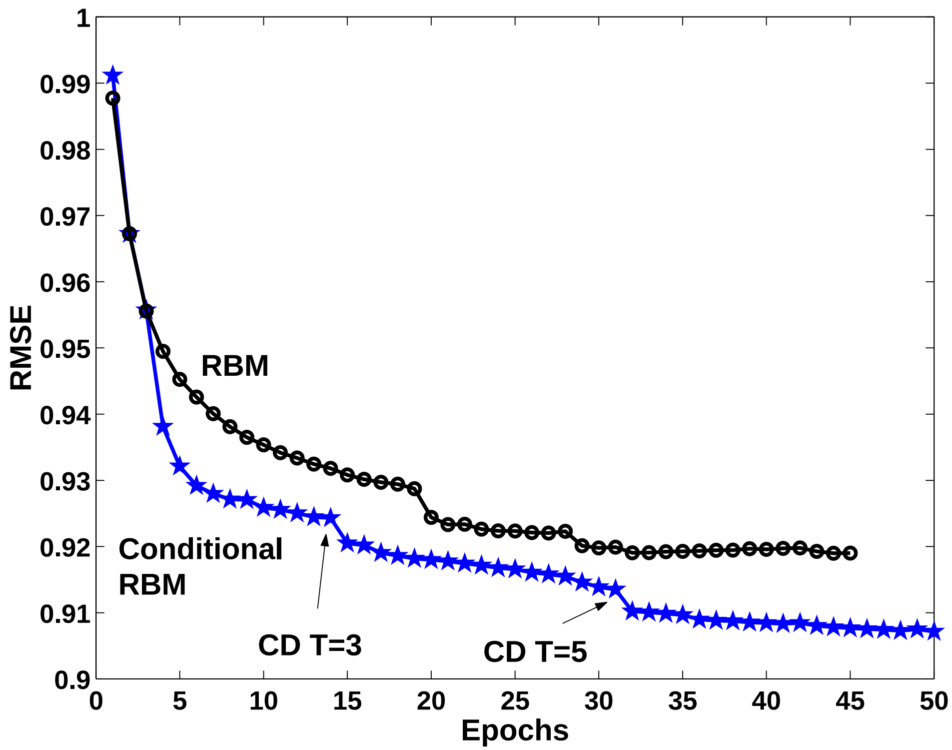
$$W_{ij}^{k} = \sum_{c=1}^{C} A_{ic}^{k} B_{cj}$$

  where typically $C \ll M$ and $C \ll F$. Setting $C = 30$, we reduce the number of free parameters by a factor of three.

- Learning matrices A and B is quite similar to learning W.

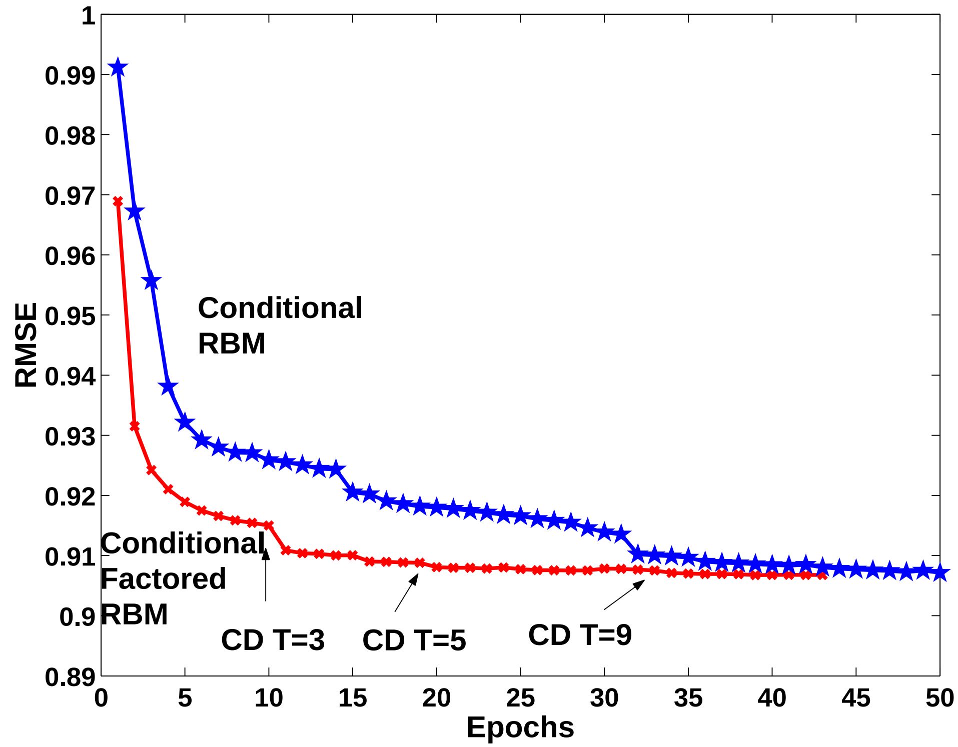
Results





- Performance of various models on the validation data.
- Left panel: RBM vs. RBM with Gaussian hidden units. Right panel: RBM vs. conditional RBM.
- The y-axis displays RMSE (root mean squared error), and the x-axis shows the number of epochs, or passes through the entire training dataset.

Results



- Performance of conditional RBM vs. conditional factored RBM
- The y-axis displays RMSE (root mean squared error), and the x-axis shows the number of epochs, or passes through the entire training dataset.

Regularized SVD

- Suppose we have M movies and N users.

- Let $X = UV'$, where $U \in R^{N\times C}$ and $V \in R^{M\times C}$ denote the low-rank approximation to the partially observed target matrix $Y \in R^{N\times M}$.

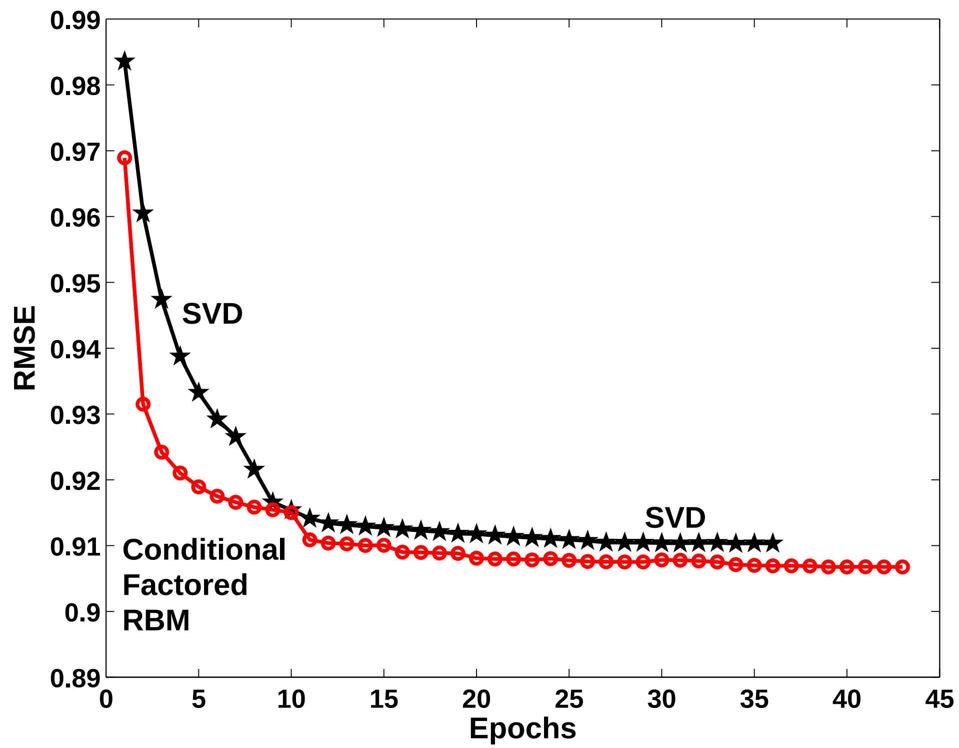
- We minimize the following objective function:

$$f = \sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}\big(\mathbf{u_i}\mathbf{v_j}' - Y_{ij}\big)^2 + \lambda \sum_{ij} I_{ij}\big(\parallel \mathbf{u_i} \parallel_{Fro}^2 + \parallel \mathbf{v_j} \parallel_{Fro}^2\big)$$

  where $\parallel \cdot \parallel_{Fro}^2$ denotes the Frobenius norm, and I_{ij} is the indicator function, taking on value 1 if user i rated movie j, and 0 otherwise.

- Unobserved entries of Y are then predicted using the corresponding entries of X.

Results



- Both models could potentially be improved by more careful tuning of learning rates, batch sizes, and weight-decay.

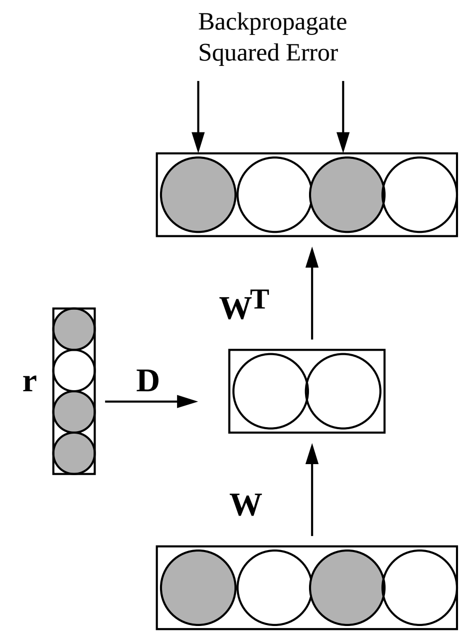
- When the predictions of multiple RBM models and multiple regularized SVD models are linearly combined, we achieve an error rate 0.8875, which is 6.72% improvement over the Netflix's own score.

Future extensions

Learning Autoencoders:

- Treat RBM learning as a *pretraining stage* that finds a good region of the parameter space.

- After pretraining, the RBM is "unrolled" to create an autoencoder network



Learning Deep Belief Nets:

- Train a stack of RBM's each having only one layer of latent (hidden) feature detectors.

- The learned feature activations of one RBM are used as the "data" for training the next layer RBM.

- This training can be used to learn a deep, hierarchical model in which each layer of features captures strong high-order correlations between the activities of features in the layer below.

THE END