Recommendation Systems theory

A/Prof Richard Yi Da Xu, Jiang Shuai
http://richardxu.com

University of Technology Sydney (UTS)

June 27, 2018

What is a Recommendation System?

- A hypothetical example of an online survey asking people to give rating of M movies with a score $1-5$:

	$Item_1$	$Item_2$	$Item_3$	$Item_4$	$Item_5$	$Item_6$	$\dots$	$Item_{M-1}$	$Item_M$
User 1	0	5	0	0	0	0	$\dots$	0	0
User 2	0	0	1	0	0	0	$\dots$	0	0
User 3	1	4	0	0	0	0	$\dots$	0	0
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
User N	0	0	5	0	0	0	$\dots$	0	0

- **zeros** doesn't mean a zero score, it means the User has not scored this public service yet.
- Extremely sparse and very large Utility matrix
- In most literature, Columns called "User" and Rows are called "Items"
- **The question is what would the score be, if the User is to score these zero entries.**

The previous example is too futuristic, so let's get back to the movie and rating example from now:

For example, **User 101** has the following rating:

	$Item_1$	$Item_2$	$Item_3$	$Item_4$	$Item_5$	$Item_6$	$\dots$	$Item_{M-1}$	$Item_M$
User 101	0	**5**	0	0	**3**	0	$\dots$	**2**	0

- **User 101** has ONLY rated three items ($Item_2 = 5$), ($Item_5 = 3$) and ($Item_{M-1} = 2$)
- From these existing ratings, system needs to decide **"recommended"** ratings for the rest $M - 3$ items
- The question is how does $Item_2$, $Item_5$ and $Item_{M-1}$ each contribute to these decisions?

Needless to say **statistics from ALL users** needed for recommendation decision for **individual User**
In Collaborative Filtering, for each pair of items (x, y):

- First obtain statistics $r_{x,y}$, for example:

	$Item_{56}$	$Item_{78}$
User 102	1	5
User 202	2	5
User 376	5	1
...	...	
User 2121	4	1

	$Item_{56}$	$Item_{78}$
User 2321	4	5
User 1232	4	4
User 3533	1	1
...	...	
User 8839	5	4

- Then compute $S_{x,y}$, which similarity measure between item x and y.
- Then recommendation for each item becomes the weighted average of these similarities measures

Pearson correlation similarity of ratings:

$$S_{x,y} = \frac{\sum_{i \in I_{xy}} (r_{x,i} - \bar{r_x})(r_{y,i} - \bar{r_y})}{\sqrt{\sum_{i \in I_{xy}} (r_{x,i} - \bar{r_x})^2 \sum_{i \in I_{xy}} (r_{y,i} - \bar{r_y})^2}}$$

cosine-based approach of ratings:

$$S_{x,y} = \frac{\sum_{i \in I_{xy}} r_{x,i} r_{y,i}}{\sqrt{\sum_{i \in I_x} r_{x,i}^2} \sqrt{\sum_{i \in I_y} r_{y,i}^2}}$$

	$Item_1$	$Item_2$	$Item_3$	$Item_4$	$Item_5$	$Item_6$	$\dots$	$Item_{M-1}$	$Item_M$
User 101	0	**5**	0	0	**3**	0	$\dots$	**2**	0

- Weighted average of these contributions is then applied
- Sometimes, clustering of users may be needed and recommendation is user-group specific. For example, Netflix users.

Recommendation System: what if it's not "ratings", but "counts"?

- Another hypothetical example of number of "views" people looking at the VET Users :

	$Course_1$	$Course_2$	$Course_3$	$Course_4$	$Course_5$	$Course_6$	...	$Course_{M-1}$	$Course_M$
student 1	0	5	0	0	0	0	...	0	0
student 2	0	0	16	0	32	0	...	0	0
student 3	1	4	0	0	0	0	...	0	0
...	...	...	...	...	...	...	...	...	...
student N	0	0	5	0	0	0	...	0	0

- The counts are unbounded.
- "Ratings of 1" means negativity rating, but "Views of 1" does NOT necessarily mean negativity.
- Negative correlation doesn't make sense; We only have "how strong" the positive correlation is.
- Recently latent Poisson Model may be used.

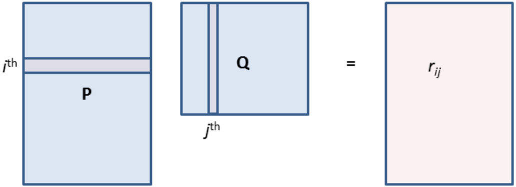
Content-based recommendations with Poisson factorization

An example of a probabilistic approach: (Gopalan, Charlin, Blei, 2014):

- Draw Item intensities $\theta_{dk} \sim \text{Gamma}(c, d)$
- Draw User preferences $\eta_{uk} \sim \text{Gamma}(e, f)$
- Draw Item topic offsets $\epsilon_{dk} \sim \text{Gamma}(g, h)$
- Draw $r_{ud} \sim \text{Poisson}(\eta_u^\top(\theta_d + \epsilon_d))$.

Recommendation System: Matrix factorisation approach, why it works?

$$\mathbf{R} \approx \mathbf{P} \times \mathbf{Q}^T = \hat{\mathbf{R}} \qquad \hat{r}_{ij} = p_i^T q_j = \sum_{k=1}^{K} p_{ik} q_{kj}$$



- number of columns of P and number of rows of Q must **agree**. However, this number K is somewhat arbitrary.
- each row of a user matrix represent a latent "user" feature vector
- each column of a item matrix represent a latent "item" feature vector
- In words, try to find matrices **P** and **Q**, such that when they multiply together the **existing ratings** have minimum changes
- The rest of zeros are replaced by non-zero numbers through matrix multiplication (think about why)
- See demo

Objective function in Matrix factorisation

- **The objective function**: what are we try to minimise?
- We just said in the previous slide that, "such that when they multiply together the **existing ratings** have minimum changes":

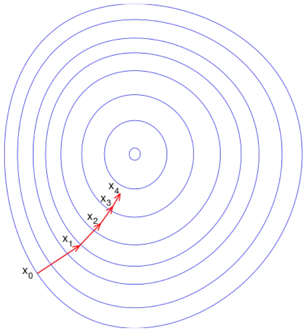
$$e_{ij}^2 = (r_{ij} - \hat{r}_{ij})^2 = \left(r_{ij} - \sum_{k=1}^{K} p_{ik} q_{kj} \right)^2 \qquad E = \sum {e_{ij}}^2 = \sum \left(r_{ij} - \sum_{k=1}^{K} p_{ik} q_{kj} \right)^2$$

We want to find all $\{p_{ik}\}$ and $\{q_{kj}\}$ which minimize E

- Note that $\arg\min(p_{ik})$ depends on one row of $\mathbf{P}$ and one column of $\mathbf{Q}$.
- We can't just let every $\frac{\partial}{\partial p_{ik}} e_{ij}^2 = 0$ and solve them at once.
- We need iterative algorithm, called **Gradient Descent** - and let's take a look:

Gradient Descend in matrix factorisation

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \alpha_n \nabla f(\mathbf{x}_n), \ n \geq 0$$



In the case of recommendation system, we have (remember **Chain rule** from high school?)

$$\frac{\partial}{\partial p_{ik}} e_{ij}^2 = -2(r_{ij} - \hat{r}_{ij})(q_{kj}) = -2e_{ij}q_{kj}$$
$$\frac{\partial}{\partial q_{kj}} e_{ij}^2 = -2(r_{ij} - \hat{r}_{ij})(p_{ik}) = -2e_{ij}p_{kj}$$

$$\begin{aligned} p'_{ik} &= p_{ik} - \alpha_n \underbrace{(-2e_{ij}q_{kj})}_{\nabla f(\mathbf{x}_n)} \\ &= p_{ik} + \alpha_n (2e_{ij}q_{kj}) \end{aligned}$$

$$\begin{aligned} q'_{kj} &= q_{kj} - \alpha_n \underbrace{(-2e_{ij}p_{kj})}_{\nabla f(\mathbf{x}_n)} \\ &= q_{kj} + \alpha_n (2e_{ij}p_{ik}) \end{aligned}$$

Recommendation System: A Matrix factorization approach (3)

- There is this so-called, "identifiability" problem in solving $\arg\min_{A,B} f(AB)$
- Hence let's put a "regulariser" and obtain a new objective function for e_{ij}

$$e_{ij}^2 = (r_{ij} - \sum_{k=1}^{K} p_{ik} q_{kj})^2 + \frac{\beta}{2} \sum_{k=1}^{K} (||P||^2 + ||Q||^2)$$

- Then, the new gradient descent algorithm becomes that of:

$$p'_{ik} = p_{ik} + \alpha \frac{\partial}{\partial p_{ik}} e_{ij}^2 = p_{ik} + \alpha(2 e_{ij} q_{kj} - \beta p_{ik})$$

$$q'_{kj} = q_{kj} + \alpha \frac{\partial}{\partial q_{kj}} e_{ij}^2 = q_{kj} + \alpha(2 e_{ij} p_{ik} - \beta q_{kj})$$

- An important extension is the requirement that all the elements of the factor matrices $\mathbf{P}$ and $\mathbf{Q}$ should be non-negative.
- Some of my researches are to add **prior probabilities** to the factor matrix, not only make them non-negative, but also enjoy other properties, such as sparsity etc.
- How we choose the optimal K? A lot of my research is in this area.
- Cold Start Problem - where no rating has been given by the user - clustering helps.
- One thing to note is that matrix factorization is very computational expensive. Stochastic Gradient Descent methods are used recently
- **Stochastic** is a buzz word of machine learning in BIG DATA era.

Ordinary least squares

- In Ordinary Least Squares (OLS) **without** regulariser, we solve for β by minimizing the squared error $||y - X\beta||_2$:

$$\textbf{Solution} \qquad \beta = \left(X^T X\right)^{-1} X^T y$$

- In Ordinary Least Squares (OLS) **with** regulariser, we solve for β by minimizing the squared error $||y - X\beta||_2 + \lambda||\beta||_2$:

$$\textbf{Solution} \qquad \beta = \left(X^T X + \lambda I\right)^{-1} X^T y$$

Alternating least squares

$$\beta^* = \arg\max_{\beta} \left(\|y - X\beta\|_2 + \lambda \|\beta\|_2 \right) \implies \beta = \left(X^T X + \lambda I \right)^{-1} X^T y$$

- If we fix Q and optimize for P alone, the problem reduced to linear regression:

$$\forall p_i : J(p_i) = \|R_i - p_i Q^T\|_2 + \lambda \cdot \|p_i\|_2$$
$$\forall q_j : J(q_j) = \|R_j - P q_j^T\|_2 + \lambda \cdot \|q_j\|_2$$

  Matching solutions for p_i and q_j are:

$$p_i = \left(Q^T Q + \lambda I \right)^{-1} Q^T R_i$$
$$q_j = \left(P^T P + \lambda I \right)^{-1} P^T R_j$$

- Since each p_i doesn't depend on other $p_{j \neq i}$, each step can potentially be introduced to massive parallelization.

Bounded approach to NNMF

- In here, we want to assign similarities, i.e., $(-1, \dots 1)$ in each entry:

	$Item_1$	$Item_2$	$Item_3$	$Item_4$	$Item_5$	$Item_6$	$\dots$	$Item_{M-1}$	$Item_M$
User 1	0	0.6	0	0	0.4	0	$\dots$	0	0
User 2	0	0.9	0.3	0.2?	0	0.5	$\dots$	0	0
User 3	0.1	0.4?	0.2	0	0.7	0	$\dots$	0.2	0
User 4	0	?	0	?	0	0	$\dots$	0	0
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
User N	0.5	0	0.6	0	0	0	$\dots$	0	0

- This is part of our **new** research
- We can also set the upper bound to each of the ratings (think about why this is useful?)

Bounded approach to NNMF: Taking in the Popularities

- Looking at the following "viewing" scores:

	$Item_1$	$Item_2$	$Item_3$	$Item_4$	$Item_5$	$Item_6$	$\dots$	$Item_{M-1}$	$Item_M$
User 1	3	0	15	0	4	0	$\dots$	6	0
User 2	12	24	20	0	0	0	$\dots$	0	0
User 3	1	3	12	0	7	0	$\dots$	2	0
User 4	0	1	0	1	0	0	$\dots$	0	0
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
User N	5	0	6	0	0	0	$\dots$	0	0

- Some items are just **popular**!
- And some users may tend to have **lot of views**
- So can we create individual bounds for each (user, item) pairs?

Factorization Machines

Feature vector **x**

	User				Movie					Other Movies rated					Time	Last Movie rated					Target y	
	A	B	C	...	TI	NH	SW	ST	...	TI	NH	SW	ST	...	Time	TI	NH	SW	ST	...		
$\mathbf{x}^{(1)}$	1	0	0	...	1	0	0	0	...	0.3	0.3	0.3	0	...	13	0	0	0	0	...	5	$y^{(1)}$
$\mathbf{x}^{(2)}$	1	0	0	...	0	1	0	0	...	0.3	0.3	0.3	0	...	14	1	0	0	0	...	3	$y^{(2)}$
$\mathbf{x}^{(3)}$	1	0	0	...	0	0	1	0	...	0.3	0.3	0.3	0	...	16	0	1	0	0	...	1	$y^{(2)}$
$\mathbf{x}^{(4)}$	0	1	0	...	0	0	1	0	...	0	0	0.5	0.5	...	5	0	0	0	0	...	4	$y^{(3)}$
$\mathbf{x}^{(5)}$	0	1	0	...	0	0	0	1	...	0	0	0.5	0.5	...	8	0	0	1	0	...	5	$y^{(4)}$
$\mathbf{x}^{(6)}$	0	0	1	...	1	0	0	0	...	0.5	0	0.5	0	...	9	0	0	0	0	...	1	$y^{(5)}$
$\mathbf{x}^{(7)}$	0	0	1	...	0	0	1	0	...	0.5	0	0.5	0	...	12	1	0	0	0	...	5	$y^{(6)}$

$$
\begin{aligned}
\hat{y}(\mathbf{x}) &= w_0 + \sum_{i}^{n} w_i x_i + \mathbf{x}^\top \mathrm{triu}(\mathbf{W})\mathbf{x} \\
&= w_0 + \sum_{i}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \mathbf{W}_{i,j} x_i x_j \\
&= w_0 + \sum_{i}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j
\end{aligned}
$$

Some computation-efficient factor

$$
\begin{aligned}
&\sum_{i}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j \\
=& \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j - \frac{1}{2} \sum_{i=1}^{n} \langle \mathbf{v}_i, \mathbf{v}_i \rangle x_i x_i \\
=& \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{f=1}^{k} v_{i,f} v_{j,f} x_i x_j - \frac{1}{2} \sum_{i=1}^{n} \sum_{f=1}^{k} v_{i,f} v_{i,f} x_i x_i \\
=& \frac{1}{2} \sum_{f=1}^{k} \left(\sum_{i=1}^{n} \sum_{j=1}^{n} v_{i,f} v_{j,f} x_i x_j - \sum_{i=1}^{n} v_{i,f} v_{i,f} x_i x_i \right) \\
=& \frac{1}{2} \sum_{f=1}^{k} \left(\left(\sum_{j=1}^{n} v_{j,f} x_j \right) \left(\sum_{i=1}^{n} v_{i,f} x_i \right) - \sum_{i=1}^{n} \left(v_{i,f} x_i \right)^2 \right) \\
=& \frac{1}{2} \sum_{f=1}^{k} \left(\left(\sum_{i=1}^{n} v_{i,f} x_i \right)^2 - \sum_{i=1}^{n} \left(v_{i,f} x_i \right)^2 \right)
\end{aligned}
$$

computational complexity is $O(kn)$

Faster NNMF convergence: Multiplicative Update Rule

- NNMF using Gradient Descend can be prohibitively slow when matrix is large
- A much faster (convergence) approach is to use "**Multiplicative Update Rule**".
- A "nature" publication and popular since Year 2000.

Faster NNMF convergence: Multiplicative Update Rule

- **Apologies** for the notations (this is to inline with each paper) $P \rightarrow W$ and $Q \rightarrow H$
- **Task:** Minimize $\|V - WH\|_2$ with respect to W and H, subject to the constraints $W, H \geq 0$.
- The Euclidean distance $\|V - WH\|$ is non-increasing under the update rules:

$$H_{a\mu} \leftarrow H_{a\mu} \frac{(W^\top V)_{a\mu}}{(W^\top WH)_{a\mu}} \qquad W_{ia} \leftarrow W_{ia} \frac{(VH^\top)_{ia}}{(WHH^\top)_{ia}}$$

- It looks so easier, but why this update rule works?

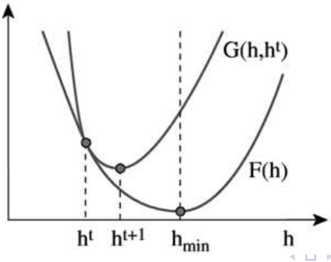
Multiplicative Update Rule

- Let's assume it's **hard** to minimize $F(h)$
- and it's easier to minimize $G(h, h^t)$. Let's find some **auxiliary function** $G(h, h^t)$ s.t.,:

$$G(h, h^t) \geq F(h), \qquad G(h, h) = F(h)$$

$$\text{Let} \qquad h^{t+1} = \arg\min_h G(h, h^t)$$

$$F(h^t) = G(h^t, h^t) \geq \underbrace{G(h^{t+1}, h^t) \geq F(h^{t+1})}_{\text{true for all } h \text{ include } h^{t+1}}$$



Faster NNMF convergence

- How are we going to prove:

$$F(h^t) = G(h^t, h^t) \geq G(h^{t+1}, h^t) \geq F(h^{t+1})$$

- $F(h^t)$ in the context of non-negative matrix factorization is:

$$\begin{aligned}
F(h) &= \frac{1}{2}\|v - Wh\|^2 \\
&= \frac{1}{2}\left(v^\top v - v^\top Wh - h^\top W^\top v + h^\top W^\top Wh\right) = \frac{1}{2}\left(v^\top v - 2v^\top Wh + h^\top W^\top Wh\right) \\
&\text{where } \nabla F(h) = W^\top Wh - W^\top v \\
&= F(h^t) + (h - h^t)^\top \nabla F(h^t) + \frac{1}{2}(h - h^t)^\top \underline{(W^\top W)}(h - h^t) \qquad \text{taylor expansion} \\
G(h, h^t) &= F(h^t) + (h - h^t)^\top \nabla F(h^t) + \frac{1}{2}(h - h^t)^\top \underline{K(h^t)}(h - h^t) \\
&\text{where } K_{a,b}(h^t) = \frac{\delta_{a,b}(W^\top Wh^t)_a}{h^t_a}
\end{aligned}$$

$$
\begin{aligned}
G(h, h^t) \geq F(h) \implies & \frac{1}{2}(h - h^t)^\top \underline{K(h^t)}(h - h^t) \geq \frac{1}{2}(h - h^t)^\top \underline{(W^\top W)}(h - h^t) \geq 0 \\
\implies & \frac{1}{2}(h - h^t)^\top \big(K(h^t) - W^\top W\big)(h - h^t) \geq 0 \\
\implies & \big(K(h^t) - W^\top W\big) \text{ is a positive definite matrix } \textbf{need to prove it}
\end{aligned}
$$

- At each iteration, we just need to find: we simplify $K(h)$ with K:

$$
\begin{aligned}
G(h, h^t) =& F(h^t) + (h - h^t)^\top \nabla F(h^t) + \frac{1}{2}(h - h^t)^\top K(h - h^t) \\
=& F(h^t) + (h - h^t)^\top \nabla F(h^t) + \frac{1}{2}(h^\top Kh \underbrace{-h^{t\top} Kh - h^\top Kh^t}_{=-2h^\top Kh^t} + h^{t\top} Kh^t)
\end{aligned}
$$

$$
\begin{aligned}
\nabla G(h, h^t) &= \nabla F(h^t) + Kh - Kh^t = 0 \\
\implies Kh &= Kh^t - \nabla F(h^t) \\
h &= h^t - K^{-1} \nabla F(h^t)
\end{aligned}
$$

- writing it properly:

$$
h^{(t+1)} \leftarrow h^t - K^{-1}(h^t) \nabla F(h^t)
$$

- We need to put the following:

$$h^{(t+1)} \leftarrow h^t - K^{-1}(h^t)\nabla F(h^t)$$

in the form of:

$$H_{a\mu} \leftarrow H_{a\mu}\frac{(W^\top V)_{a\mu}}{(W^\top WV)_{a\mu}} \qquad \text{or} \qquad h_a \leftarrow h_a\frac{(W^\top V)_a}{(W^\top WV)_a}$$

$$\begin{aligned} K_{a,b}(h^t) = \frac{\delta_{a,b}(W^\top Wh^t)_a}{h^t_a} &\implies K(h^t) = \begin{bmatrix} \frac{(W^\top Wh^t)_1}{h^t_1} & \dots \\ \dots & \frac{(W^\top Wh^t)_N}{h^t_N} \end{bmatrix} \\ &\implies K^{-1}(h^t) = \begin{bmatrix} \frac{h^t_1}{(W^\top Wh^t)_1} & \dots \\ \dots & \frac{h^t_N}{(W^\top Wh^t)_N} \end{bmatrix} \end{aligned}$$

Faster NNMF convergence

- therefore,

$$
\begin{aligned}
h_a^t - \Big(K^{-1}(h^t) \underbrace{\nabla F(h^t)}_{W^\top Wh - W^\top v}\Big)_a &= h_a^t - \frac{h_a^t}{(W^\top Wh^t)_a}(W^\top Wh^t - W^\top v)_a \\
&= h_a^t - \frac{h_a^t(W^\top Wh^t - W^\top v)_a}{(W^\top Wh^t)_a} \\
&= \frac{h_a^t(W^\top Wh^t)_a - h_a^t(W^\top Wh^t)_a - h_a^t(W^\top v)_a}{(W^\top Wh^t)_a} \\
&= h_a^t \frac{(W^\top v)_a}{(W^\top Wh^t)_a}
\end{aligned}
$$

- One can obtain update for W in a similar fashion.

Lastly, how do we know $\left(K(h^t) - W^\top W\right)$ is a positive definite matrix?

$$K_{a,b}(h^t) = \frac{\delta_{a,b}(W^\top W h^t)_a}{h_a^t} = \frac{\delta_{a,b}\sum_i (W^\top W)_{a,i} h_i^t}{h_a^t}$$

Therefore,

$$\begin{aligned}
&\sum_{a,b} v_a \left[h_a^t K_{a,b}(h^t) h_b^t\right] v_b \\
&= \sum_{a,b} v_a h_a^t \left(\frac{\delta_{a,b}\sum_i (W^\top W)_{a,i} h_i^t}{h_a^t}\right) h_b^t v_b \\
&= \sum_{a} v_a h_a^t \left(\frac{\sum_i (W^\top W)_{a,i} h_i^t}{h_a^t}\right) h_a^t v_a \\
&= \sum_{a} \left(\sum_i (W^\top W)_{a,i} h_i^t\right) h_a^t v_a^2 \\
&= \sum_{a,b} (W^\top W)_{a,b} h_b^t h_a^t v_a^2
\end{aligned}$$

Lastly, how do we know $\left(K(h^t) - W^\top W\right)$ is a positive definite matrix?

$$
\begin{aligned}
v^\top M v &= \sum_{ab} v_a M_{a,b}(h^t) v_b = \sum_{a,b} v_a \underbrace{\left[h_a^t (K(h^t) - W^\top W)_{a,b} h_b^t\right]}_{M_{a,b}(h^t)} v_b \\
&= \sum_{a,b} v_a \left[h_a^t K_{a,b}(h^t) h_b^t\right] v_b - \sum_{a,b} v_a \left[h_a^t (W^\top W)_{a,b} h_b^t\right] v_b \\
&= \sum_{a,b} \left[(W^\top W)_{a,b} h_b^t h_a^t v_a^2\right] - \left[v_a h_a^t (W^\top W)_{a,b} h_b^t v_b\right] \qquad \text{see previous slide} \\
&= \sum_{a,b} (W^\top W)_{a,b} h_a^t h_b^t \left[v_a^2 - v_a v_b\right] \\
&= \frac{1}{2}\left(\sum_{a,b} (W^\top W)_{a,b} h_a^t h_b^t \left[v_a^2 - v_a v_b\right] + (W^\top W)_{a,b} h_a^t h_b^t \left[v_a^2 - v_a v_b\right]\right) \\
&= \frac{1}{2}\left(\sum_{a,b} (W^\top W)_{a,b} h_a^t h_b^t \left[v_a^2 - v_a v_b\right] + \underbrace{(W^\top W)_{b,a} h_b^t h_a^t \left[v_b^2 - v_b v_a\right]}_{\text{swap } a \text{ and } b}\right) \\
&= \frac{1}{2}\left(\sum_{a,b} (W^\top W)_{a,b} h_a^t h_b^t \left[v_a^2 - v_a v_b\right] + (W^\top W)_{b,a} h_b^t h_a^t \left[v_b^2 - v_b v_a\right]\right) \\
&= \frac{1}{2}\left(\sum_{a,b} (W^\top W)_{a,b} h_a^t h_b^t \left[v_a^2 - v_a v_b + v_b^2 - v_b v_a\right]\right) \\
&= \frac{1}{2} \sum_{a,b} (W^\top W)_{a,b} h_a^t h_b^t \left[v_a - v_b\right]^2 \qquad \text{since } W,\ h^t \text{ are all non-negative}
\end{aligned}
$$