Appendix

1. Collaborative filtering of latent factors using singular value decomposition (SVD)

For our matrix factorization model we use a form of SVD called *Funk SVD* as popularized by Simon Funk [1], for transforming our input matrix into the latent factor model. SVD decomposes an input matrix X, where rows represent workload and columns represent the resource configurations, into the product of two matrices, U and V^T as described by Equation 1. Each element x_{nm} of X represents the *normalized exec_time* (performance value) of executing workload n on configuration m. As most (and perhaps all) serverless platforms have a maximum execution time limit, we use the *min-max* normalization.

$$\begin{aligned} X_{N\times M} &= U_{N\times K} \times V^T_{K\times M} \\ x_{n,m} &= u_n^T v_m = \sum_1^K u_{n,k} v_{m,k} \end{aligned} \tag{1}$$

We start by filling in the missing entries of X by random initialization. Next, we minimize the *loss* of U and V to obtain $u_n^T v_m \approx x_{n,m}; \forall n, m$. The loss function is the root mean square error (RSME) described in Equation 2, which is minimized using the learning algorithms.

$$RMSE = \sqrt{\frac{1}{n}\sum_{(u,v)} |(x_{u,v} - p_{u,v})^2|} \tag{2}$$

where $p_{u,v}$ and $x_{u,v}$ are the predicted and observed performance values for workload u and configuration v respectively. Here, the predicted values p for the missing entries is computed by the following Equation 3:

$$p_{n,m} = u_n^T v_m = \sum_1^K u_{n,k} v_{m,k} \tag{3}$$

Thus, we can formulate the objective function for learning the factor vectors (u_n, v_m) as,

$$\mathcal{L} = \sum_{(m,n)\in\Omega} (x_{n,m} - u_n^T v_m)^2 \tag{4}$$

To avoid overfitting, a common technique is to use Tikhonov regularization [2] to transform the low-rank approximation problem into the following regularized objective function 5:

$$\mathcal{L}^{reg} = \sum_{(m,n)\in\Omega} (x_{n,m} - u_n^T v_m)^2 + \lambda_u ||u_n||_2^2 + \lambda_v ||v_m||_2^2 \tag{5}$$

We considered two widely used learning algorithms to minimize this objective function.

Stochastic gradient descent (SGD): In the SGD approach, we start by randomly initializing the missing entries and then run SGD to update these unknown entries using an objective function that minimizes the mean squared error on the known entries of the matrix. SGD computes a single prediction p_{nm} using Equation 3 and it's error:

$$\begin{aligned} e_{nm} &= (x_{nm} - p_{nm}) \\ e_{nm} &= (x_{nm} - u_n^T v_m) \end{aligned} \tag{6}$$

The parameters of this algorithm are updated by a magnitude proportional to the learning rate η in the opposite direction of the gradient resulting in the following update rules [3] as described in the Equations 7 and 8:

$$u_n = u_n + \eta(e_{nm} v_m - \lambda_u u_n) \tag{7}$$

$$v_m = v_m + \eta(e_{nm} u_n - \lambda_v v_m) \tag{8}$$

Alternating least squares (ALS): ALS takes a closed-form approach to optimizing the loss function. The key insight is that you can turn the non-convex optimization problem in Equation 1 into an "easy" quadratic problem that can be solved optimally if you fix either u_n or v_m [3]. When one is fixed the other can be computed by solving the least-squares problem (Equations 9 and 10) [2], [4].

$$argmin_{v_m} \sum_{n\in\Omega_{c_m}} (x_{n,m} - u_n^T v_m)^2 + \lambda_v ||v_m||_2^2 \tag{9}$$

$$argmin_{u_n} \sum_{m\in\Omega_{r_n}} (x_{n,m} - u_n^T v_m)^2 + \lambda_u ||u_n||_2^2 \tag{10}$$

From this, matrix factorization can be seen as a sequence of regression problems (one for each latent factor) by showing that equations 9 and 10 have the following solution described by equations 11 and 12.

$$v_m^* = \left(\sum_{n\in\Omega_{c_m}} u_n u_n^T + \lambda_v I_K\right)^{-1} \sum_{n\in\Omega_{c_m}} x_{n,m} u_n \tag{11}$$

$$u_n^* = \left(\sum_{m\in\Omega_{r_n}} v_m v_m^T + \lambda_u I_K\right)^{-1} \sum_{m\in\Omega_{r_n}} x_{n,m} v_m \tag{12}$$

2. Active learning strategies used in our collaborative filtering

Variance: This strategy selects the rows with the highest variance, hence, it favours the workloads that have performed diversly across the configurations on the assumption that the variance gives an indication of the uncertainty of the system about that workload's performance

$$Variance(i) = \frac{1}{|U_i|} \sum_{u \in U_i} (r_{ui} - \overline{r}_i)^2 \tag{13}$$

where U_i is the set of workloads executed on config. i, and $\overline{r}_i$ is the average value for i.

Similarity: This approach selects the rows with the highest similarity to the jobs previously used configs.

$$sim(u, v) = \frac{\sum\limits_{i \in I_{uv}} (r_{ui} - \overline{r}_u)(r_{vi} - \overline{r}_v)}{\sqrt{\sum\limits_{i \in I_{uv}} (r_{ui} - \overline{r}_u)^2 \sum\limits_{i \in I_{uv}} (r_{vi} - \overline{r}_v)^2}} \tag{14}$$

Here $\overline{r}_u$ denotes the average *perf_value* of job u, $sim(u, v)$ is the similarity of the jobs u and v and I_{uv} is the set of configs co-used by workloads u and v. The job-to-job similarity can be computed using different approaches. A popular one is Pearson correlation [5], [6].

References

[1] Simon Funk. Netflix update: Try this at home. https://sifter.org/~simon/journal/20061211.html, December 2006.

[2] Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. Large-scale parallel collaborative filtering for the netflix prize. In *International conference on algorithmic applications in management*, pages 337–348. Springer, 2008.

[3] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[4] Hsiang-Fu Yu, Cho-Jui Hsieh, Si Si, and Inderjit S Dhillon. Parallel matrix factorization for recommender systems. *Knowledge and Information Systems*, 41(3):793–819, 2014.

[5] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186, 1994.

[6] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295, 2001.