Cross-Domain Scruffy Inference

Kenneth C. Arnold
MIT Media Lab and MIT Mind Machine Project
Cambridge, MA 02139, USA
kcarnold@media.mit.edu

Henry Lieberman
MIT Media Lab and MIT Mind Machine Project
Cambridge, MA 02139, USA
lieber@media.mit.edu

Abstract

Reasoning about Commonsense knowledge poses many problems that traditional logical inference doesn't handle well. Among these is cross-domain inference: how to draw on multiple independently produced knowledge bases. Since knowledge bases may not have the same vocabulary, level of detail, or accuracy, that inference should be "scruffy." The AnalogySpace technique showed that a factored inference approach is useful for approximate reasoning over noisy knowledge bases like ConceptNet. A straightforward extension of factored inference to multiple datasets, called Blending, has seen productive use for commonsense reasoning. We show that Blending is a kind of Collective Matrix Factorization (CMF): the factorization spreads out the prediction loss between each dataset. We then show that blending additional data causes the singular vectors to rotate between the two domains, which enables cross-domain inference. We show, in a simplified example, that the maximum interaction occurs when the magnitudes (as defined by the largest singular values) of the two matrices are equal, confirming previous empirical conclusions. Finally, we describe and mathematically justify Bridge Blending, which facilitates inference between datasets by specifically adding knowledge that "bridges" between the two, in terms of CMF.

Introduction

Factored Inference

Inference on imprecise assertional knowledge is difficult. Logical inference requires the knowledge base to be consistent, assertions to be accurately qualified, and entities and relations carefully specified. Bayesian inference requires that the conditional probability distributions relating observations be well understood or learned from large quantities of data. Knowledge bases contributed by humans or mined from free text do not readily meet such requirements (though steps have been taken to ease them). AnalogySpace (Speer, Havasi, and Lieberman 2008) uses a new kind of reasoning, which we call *factored inference*, or loosely (after the old neats-vs-scruffies dichotomy in AI research styles), "scruffy inference."

Factored inference considers a matrix A of all possible assertions (which may never be computed explicitly). The matrix encodes an assertion as a numerical confidence a_{ij} that *concept* i (e.g., FOOD) has *feature* j (e.g., PERSON\Desires/___). The inference process models A as a matrix product of two much smaller matrices: $A \approx UV^T$. The dimensions of A are concepts n_c by features n_f; the dimensions of U and V are $n_c \times k$ and $n_f \times k$, where $k \ll n_c$. U and V can be viewed as simplified representations for each concept and feature: rather than representing a concept by how much each of the n_f features apply to it, U instead represents a concept by how much each of a smaller number k of characteristics applies to it. Then V defines those characteristics by how much they apply to each of the features. Since k is small relative to the total number of concepts and features, this approach is broadly termed *dimensionality reduction*.

Factored inference makes conclusions based on a simplified view of the world: it considers concepts and features through the lens of k general characteristics; if a concept and a feature look similar through those lenses, it concludes that the concept has that feature. That is, to determine if concept i has feature j, factored inference considers the similarity (usually given by the dot product) of two k-element vectors: row i of U for the concept and row j of V for the feature. This style of inference is robust in the sense that isolated spurious data does not significantly alter the overall view of the world. Though this representation has many limitations (it is incapable of deductive, rigorous, or deeply contextual reasoning in its current form), it has nonetheless been useful for broad reasoning over incomplete and inaccurate knowledge.

Cross-Domain Factored Inference

AnalogySpace first used factorization for inference over the assertions in the ConceptNet database of user-contributed natural language commonsense knowledge (Havasi, Speer, and Alonso 2007). But while scruffy reasoning within a single knowledgebase is useful, being able to reason simultaneously over data from different domains is far more useful.

Again, logical and statistical approaches to the problem of cross-domain joint inference are possible. One approach is to place all the data within a statistical model, a such as a graphical model, that treats each kind of data as evidence about one or more hidden variables. Another approach is to construct logical assertions from each dataset, adding inference rules that allow translation between different types of assertions. However, both approaches require substantial *a priori* understanding of how the data should interact (i.e.,

what graph connections are required? what are the proper inference rules?). Again, a "scruffy" approach trades precision for breadth and robustness.

Blending (Havasi et al. 2009) extends the factored inference of AnalogySpace to data from multiple domains: commonsense concepts, domain-specialized concepts (and features), and even non-linguistic entities. It has been shown to be useful for many applications, including opinion mining (Speer et al. 2010), word sense disambiguation (Havasi 2009), and goal-oriented code search (Arnold and Lieberman 2010). Where data from multiple domains refers to the same set of entities (concepts or features), the representation of those entities accounts for each dataset. So when the factored representations are used to make inferences, those inferences are influenced by all the input datasets.

How to Factor One or More Matrices

Singular Value Decomposition

One common process for factoring the matrix of assertions A is the singular value decomposition (SVD), which expresses A in terms of its singular values and singular vectors. If $A\vec{v} = \sigma\vec{u}$ and $A^T\vec{u} = \sigma\vec{v}$, for unit vectors $\vec{i}$ and $\vec{j}$, then σ is a singular value of A and $\vec{u}$ and $\vec{v}$ are the corresponding singular vectors. We can solve for the singular vectors by substitution: to solve for $\vec{u}$, multiply on the left by A:

$$AA^T\vec{u} = A\sigma\vec{v} = \sigma A\vec{u} = \sigma^2\vec{u}.$$

Similarly, $A^TA\vec{v} = \sigma^2\vec{v}$. So we see that $\vec{u}$ is an eigenvector of AA^T, with an eigenvalue of σ^2. Arrange the singular vectors $\vec{u}_i$, $\vec{v}_i$ in the columns of matrices $U = [\vec{u}_1 \mid \vec{u}_2 \mid \cdots \mid \vec{u}_n]$ and $V = [\vec{v}_1 \mid \vec{v}_2 \mid \cdots \mid \vec{v}_n]$, and arrange the singular values σ_i along the diagonal of Σ. Order them by decreasing σ and stop after k singular vectors. Now we can write the singular value decomposition of A: $A \approx U\Sigma V^T$. (We can see that this decomposition is consistent with the general factoring definition by writing $A \approx (U\sqrt{\Sigma})(V\sqrt{\Sigma})^T$.) Consider a feature $\vec{f}$, specified by how much it applies to each concept. $A\vec{f}$ gives the weighted sum of the concepts that have that feature, $\vec{c}$, specified by how much each feature applies to all those concepts. The SVD allows us to write $A\vec{f} \approx U\Sigma V^T\vec{f}$, which, read right-to-left, gives an intuitive understanding of the SVD. First, $V^T\vec{f}$ gives how much each singular vector $\vec{v}_i$ is associated with $\vec{f}$. Then, Σ scales by the prominence of each singular vector. Finally, U expresses how much each concept is associated with each singular vector.

A Bregman divergence (Singh and Gordon 2008) measures the error, under a given function F, of using $\hat{A}$ instead of A:

$$D_F(\hat{A}|A) = \sum_{ij} F(a_{ij}) - F'(a_{ij})\hat{a}_{ij} + F^*(F'(a_{ij})),$$

where $F^*(m)$ is the convex dual (Legendre transform) of F and gives the negative of the y-intercept of the line tangent to F with slope m, thus ensuring that the minimum divergence is 0. It can be shown that the SVD minimizes the Bregman divergence for $\hat{A} = U\Sigma V^T$ under squared loss $F(x) = x^2$:

$$D_{x^2}(\hat{A}|A) = \sum_{ij}(a_{ij} - \hat{a}_{ij})^2.$$

Factoring Two Matrices Using One Big Matrix

Suppose we have two data sources, e.g., ConceptNet and an ontology like WordNet or some domain-specific knowledge base. If they have concepts or features in common, or their concepts and features can be aligned, the Blending technique provides a way to analyze them together. It constructs a simplified view of the world, similar to that of AnalogySpace, where at least one of the "lenses" (principal components) considers characteristics from both datasets. The result can infer that if a concept has certain characteristics in one domain that it may have related characteristics in another domain.

Blending works by constructing a single large matrix A composed of weighted sums of each dataset, aligned and zero-padded so that the row and column labels match.[1] Reordering the rows or columns of a matrix, or adding rows or columns of all zeros, merely reorders or adds zero rows to the U and V matrices in the SVD of each input matrix.

The technique works for an arbitrary number of datasets in a variety of configurations, even overlapping. For this presentation, however, we consider only a simple blend of two matrices, X and Y. Suppose that they overlap in concepts but not features, and that this time concepts are placed in columns. Since the features are disjoint, we can write the weighted sum matrix as $A = \begin{bmatrix}(1-\alpha)X \\ \alpha Y\end{bmatrix}$. We now factor A (using the SVD, for example):

$$A = \begin{bmatrix}(1-\alpha)X \\ \alpha Y\end{bmatrix} \approx \begin{bmatrix}U_X \\ U_Y\end{bmatrix} V^T = \begin{bmatrix}U_XV^T \\ U_YV^T\end{bmatrix}.$$

This decomposition says that we are forced to use the same V to factor both X and Y. If we had factored either of X or Y alone, we would have gotten a V optimized for reconstructing the corresponding matrix. But instead, V must account (to some degree, controlled by α) for both matrices, in order to make a reasonable reconstruction.

Blending Spreads Out the Loss, Like CMF

What problem is Blending solving? The SVD and many related algorithms minimize some Bregman divergence: the error, with respect to a given function F, of using $\hat{A} = UV^T$ instead of A. In this blended setup, we have

$$D\left(\begin{bmatrix}U_XV^T \\ U_YV^T\end{bmatrix} \mid \begin{bmatrix}(1-\alpha)X \\ \alpha Y\end{bmatrix}\right).$$

If the divergence function is separable by element (which it is for the SVD's squared-error loss), we can write the divergence as instead

$$D(U_XV^T|(1-\alpha)X) + D(U_YV^T|\alpha Y).$$

So the factorization minimizes a weighted loss over both matrices. Thus, Blending is a special case of Collective Matrix Factorization (Singh and Gordon 2008), a factorization

[1] Sometimes the labels of the input data may not match exactly; for example, WordNet distinguishes word senses, whereas ConceptNet does not. In such a case, options include treating ConceptNet data as applying to all word senses equally, ignoring word sense distinctions, or using the Bridge Blending technique discussed later.

framework that generalizes to other loss functions, corresponding to different prior assumptions or constraints such as non-negativity.

At either extreme of $\alpha = 0$ or 1, one of the loss terms gets zero effective weight, so V^T is chosen purely because of one matrix. But at intermediate values of α, the loss is spread over the two matrices, illustrating that the blended view of the world is shifting to accommodate both datasets, an observation that the next section seeks to clarify.

New Data Rotates the Factorization

A factorization of X is like one view of the world. If you blend in a second matrix Y with a different view of the world, how does the resulting worldview shift? In this section, we show that in a simplified context, the axes of the worldview of X rotate in the direction of the axes of the worldview of Y by an amount controlled by the weight α and the amount of overlap between X and Y. Additionally, we verify the empirical conclusion that the maximal interaction is achieved when the matrices are weighted such that their largest singular values are equal.

For the purposes of this exploration, consider again two matrices X and Y, but this time align all of their rows and columns (zero-padding as necessary) so that both their rows and columns have matching labels. The SVD of X alone is $X \approx U\Sigma V^T$; the SVD of Y alone is $Y \approx U_y\Sigma_y V_y^T$, which we'll write instead as $Y \approx AB^T$ because it's shorter. (One choice of A and B is $A = U_y$ and $B = V_y\Sigma_y$.) If either SVD is truncated, the results may differ from this theory. Our derivation here follows mathematical results originally derived for incremental updating of an SVD (Brand 2006). We seek the SVD of the sum $X + Y = X + AB^T = U'\Sigma'V'^T$. We can write that sum in a form that looks like an SVD, though the matrices do not yet have the necessary properties:

$$X + AB^T = \begin{bmatrix} U & A \end{bmatrix} \begin{bmatrix} \Sigma & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} V & B \end{bmatrix}^T. \quad (1)$$

Our strategy will be to transform this expression until the outer matrices are unitary and the inner matrix is diagonal. Though U is orthonormal, and perhaps even A is, $\begin{bmatrix} U & A \end{bmatrix}$ probably is not orthonormal because U and A probably have some components in the same direction. To make a new SVD, we must separate out $\begin{bmatrix} U & A \end{bmatrix}$ into components that are in the direction of U and components that are orthogonal to U. (More formally, we seek $A = A_\parallel + A_\perp$, where the column-space of $A_\parallel$ is a subspace of the column space of U, and the column space of $A_\perp$ is disjoint with the column space of U.)

The projection $A_\parallel$ of the columns of A into the column-space of U is given by U^TA since U is orthonormal. Now we go back into the original space by just multiplying by U: $A_\parallel = UU^TA$ is the component of A in the direction of U. Now what's orthogonal to U is just what's left over: $A_\perp = A - UU^TA$, which we can write more compactly as $A_\perp = \left(I - UU^T\right)A$. $A_\parallel$ and $A_\perp$ have the same shape as A: $m \times n$.

Let P be an orthonormal basis for the column space of $A_\perp$. P represents all of the dimensions of A that were not already accounted for by U; it is $m \times r$, where $r = \operatorname{rank}(A_\perp)$. Since P is a basis of $A_\perp$, we can write each column of $A_\perp$ as a linear combination of columns of P: $A_\perp = PR_A$. The new matrix R_A ($r \times n$) represents where each concept is in the "leftover" space.

If $A_\perp = \left(I - UU^T\right)A$, then $A = UU^TA + PR_A = \begin{bmatrix} U & P \end{bmatrix} \begin{bmatrix} U^TA \\ R_A \end{bmatrix}$. This decomposition enables us to write the leftmost term of equation 1 as

$$\begin{bmatrix} U & A \end{bmatrix} = \begin{bmatrix} U & P \end{bmatrix} \begin{bmatrix} I & U^TA \\ 0 & R_A \end{bmatrix}$$

Now, unlike $\begin{bmatrix} U & A \end{bmatrix}$, $\begin{bmatrix} U & P \end{bmatrix}$ *is* orthonormal, so we're making progress towards the SVD. Performing the same operation on V and B, we have

$$\begin{bmatrix} V & B \end{bmatrix} = \begin{bmatrix} V & Q \end{bmatrix} \begin{bmatrix} I & V^TB \\ 0 & R_B \end{bmatrix}$$

Substituting these results back into equation 1 now yields

$$\begin{aligned} X + AB^T &= \begin{bmatrix} U & A \end{bmatrix} \begin{bmatrix} \Sigma & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} V & B \end{bmatrix}^T \\ &= \begin{bmatrix} U & P \end{bmatrix} K \begin{bmatrix} V & Q \end{bmatrix}^T \\ K &= \begin{bmatrix} I & U^TA \\ 0 & R_A \end{bmatrix} \begin{bmatrix} \Sigma & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} I & V^TB \\ 0 & R_B \end{bmatrix}^T \end{aligned}$$

We now have unitary outer matrices, but a non-diagonal inner matrix K, which simplifies to

$$K = \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} U^TA \\ R_A \end{bmatrix} \begin{bmatrix} V^TB \\ R_B \end{bmatrix}^T.$$

We can diagonalize K by taking its SVD: $K = U_K\Sigma_K V_K^T$. U_K is orthonormal, and the product of two orthonormal vectors is orthonormal, so we can finally write the SVD of $X + Y$:

$$X + Y = \left(\begin{bmatrix} U & P \end{bmatrix} U_K\right) \Sigma_K \left(\begin{bmatrix} V & Q \end{bmatrix} V_K\right)^T$$

Let's consider what this means intuitively. Starting from the SVD of X alone, we first add on the new space covered only by Y; this is expressed by P and Q. However, projecting $X + Y$ into this combined space leaves much of the data "off-axis", represented by K not being diagonal. So we use the SVD of K to construct U_K and V_K that rotate the vectors to align the space with the axes of the data.

Maximizing Interaction: Veering is Rotation

(Havasi et al. 2009) observed empirically that when blending two matrices, the maximum interaction between the two datasets was observed when corresponding singular values from the two matrices were nearly equal (see (Havasi 2009) for more detail). The effect, called *veering*, can be observed on a plot of the singular values as the relative weights between the matrices change; it appears that the singular values aim to intersect but in fact seem to repel each other (a similar

plot will be visible later as Figure 1). Why is this the case? What occurs when singular values approach each other? In this section, we use the mathematical techniques developed above to show that as the veering point is approached, the resulting singular vector actually rotates between the two singular vectors, permitting cross-domain inference.

We consider a simplified problem in order to study the veering effect in isolation. Consider two rank-1 matrices $X = \vec{u}_X \vec{v}_X^T$ and $B = \alpha \vec{u}_Y \vec{v}_Y^T$, where $\vec{u}_{\{X,Y\}}$ and $\vec{v}_{\{X,Y\}}$ are unit-magnitude column vectors with matched labels. (In the numerical examples shown later, these vectors are the first singular vectors of the `IsA` and `AtLocation` portions of ConceptNet.) The result will be independent of absolute magnitude; α represents the relative magnitude of Y compared to X. $X + Y$ could be up to rank 2; what are its singular values and vectors? Following the derivation above, we start by writing

$$\begin{aligned} X + Y &= \vec{u}_X \vec{v}_X^T + \alpha \vec{u}_Y \vec{v}_Y^T \\ &= \left[\begin{array}{cc} \vec{u}_X & \vec{u}_Y \end{array}\right] \left[\begin{array}{cc} 1 & 0 \\ 0 & \alpha \end{array}\right] \left[\begin{array}{cc} \vec{v}_X & \vec{v}_Y \end{array}\right]^T . \end{aligned} \quad (2)$$

Now we wish to find the parallel and orthogonal components of $\vec{u}_Y$ with respect to $\vec{u}_X$, which is just a projection. The parallel component was $UU^T A$ from the general derivation above, but for a single column vector it simplifies to $\vec{u}_X \vec{u}_X^T \vec{u}_Y$, which is just $\vec{a}_\parallel = \vec{u_X} (\vec{u}_X \cdot \vec{u}_Y) = \vec{u}_X \cos\theta_U$, where θ_U is the angle between the two vectors. Assuming that $0 < |\cos\theta_U| < 1$, one perpendicular component exists, given by $\vec{a}_\perp = \vec{u}_Y - \vec{u_X} \cos\theta_U$. Since $\vec{a}_\perp$ is the opposite side of a right triangle with $\vec{a}_\parallel$, its magnitude is $\sin\theta_U$. We then normalize $\vec{a}_\perp$ to get $\hat{a}_\perp = \frac{\vec{u}_Y - \vec{u_X}\cos\theta_U}{\sin\theta_U}$. Then $\vec{u}_Y = \vec{u}_X \cos\theta_U + \hat{a_\perp} \sin\theta_U$, and

$$\left[\begin{array}{cc} \vec{u}_X & \vec{u}_Y \end{array}\right] = \left[\begin{array}{cc} \vec{u}_X & \hat{a_\perp} \end{array}\right] \left[\begin{array}{cc} 1 & \cos\theta_U \\ 0 & \sin\theta_U \end{array}\right] .$$

The corresponding derivation applied to $\vec{v}_Y$ yields

$$\left[\begin{array}{cc} \vec{v}_X & \vec{v}_Y \end{array}\right] = \left[\begin{array}{cc} \vec{v}_X & \hat{b_\perp} \end{array}\right] \left[\begin{array}{cc} 1 & \cos\theta_V \\ 0 & \sin\theta_V \end{array}\right] .$$

We can now write equation 2 as

$$\begin{aligned} X + Y &= \left[\begin{array}{cc} \vec{u}_X & \vec{u}_Y \end{array}\right] \left[\begin{array}{cc} 1 & 0 \\ 0 & \alpha \end{array}\right] \left[\begin{array}{cc} \vec{v}_X & \vec{v}_Y \end{array}\right]^T \\ &= \left[\begin{array}{cc} \vec{u}_X & \hat{a_\perp} \end{array}\right] K \left[\begin{array}{cc} \vec{v}_X & \hat{b_\perp} \end{array}\right]^T , \end{aligned}$$

where the inner term K is given by

$$\begin{aligned} K &= \left[\begin{array}{cc} 1 & \cos\theta_U \\ 0 & \sin\theta_U \end{array}\right] \left[\begin{array}{cc} 1 & 0 \\ 0 & \alpha \end{array}\right] \left[\begin{array}{cc} 1 & \cos\theta_V \\ 0 & \sin\theta_V \end{array}\right]^T \\ &\quad \left[\begin{array}{cc} 1 & \cos\theta_U \\ 0 & \sin\theta_U \end{array}\right] \left[\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array}\right] \left[\begin{array}{cc} 1 & \alpha\cos\theta_V \\ 0 & \alpha\sin\theta_V \end{array}\right]^T \\ &= \left[\begin{array}{cc} 1 & 0 \\ 0 & 0 \end{array}\right] + \left[\begin{array}{c} \cos\theta_U \\ \sin\theta_U \end{array}\right] \left[\begin{array}{c} \alpha\cos\theta_V \\ \alpha\sin\theta_V \end{array}\right]^T . \end{aligned}$$

One common case is blending a matrix where only either the rows or columns overlap. If the columns do not overlap, then $\vec{v}_X$ is orthogonal to $\vec{v}_Y$, so $\theta_V = \pi/2$ (so $\cos\theta_V = 0$ and $\sin\theta_V = 1$) and K becomes:

$$K = \left[\begin{array}{cc} 1 & \cos\theta_U \\ 0 & \sin\theta_U \end{array}\right] \left[\begin{array}{cc} 1 & 0 \\ 0 & \alpha \end{array}\right] I = \left[\begin{array}{cc} 1 & \alpha\cos\theta_U \\ 0 & \alpha\sin\theta_U \end{array}\right] .$$

We diagonalize K by computing its SVD: $K = U_K \Sigma_K V_K^T$. A computer algebra system was used to find that the two singular values are

$$1/2 \sqrt{2 + 2\alpha^2 \pm 2\sqrt{1 + 2\alpha^2 + \alpha^4 - 4\alpha^2 \sin^2\theta_U}},$$

which is plotted for a several values of θ_U in Figure 1; the middle value is the angle between the `IsA` and `AtLocation` vectors in ConceptNet.

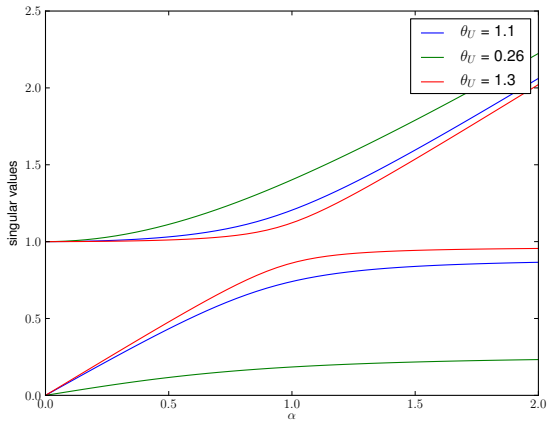


Figure 1: Singular values vs. α for several values of θ_U

Thus, the SVD of a rank-1 blend overlapping in columns only, in terms of the angle between the row vectors, is

$$\vec{u}_X \vec{v}_X^T + \alpha \vec{u}_Y \vec{v}_Y^T = \left(\left[\vec{u}_X \mid \hat{a_\perp}\right] U_K\right) \Sigma_K \left(\left[\vec{v}_X \mid \hat{b_\perp}\right] V_K\right)^T$$

A symbolic solution for the singular vectors is not as straightforward, but an intuition can be obtained by plotting how the unit circle is affected by K, remembering that U_k and V_k, being orthogonal, act as rotations about the origin. Figure 2 shows $K\vec{c}$ for $|\vec{c}| = 1$. Since $V_K^T \vec{c}$ only rotates about the origin, it does not change the circle. Then Σ stretches the circle into an ellipse along the x- and y- axes, which correspond intuitively to the X and Y matrices. Finally, U_K rotates that ellipse so that its axes align with the singular vectors $\vec{u_{k1}}$ and $\vec{u_{k2}}$, which are plotted (scaled by their singular values) in the figure.

When $\alpha = 0$, $Y = 0$; when $\theta_U = 0$, Y lies entirely within the column-space of X (in this case it is a scalar multiple of X). In either case, the ellipse is squashed onto the x-axis, indicating that only the singular vector of the result is the singular vector of X. As the magnitude α of the second matrix increases, or the angle θ_U between the two matrices increases, the first singular vector begins to rotate off of the x-axis, indicating that it is now a linear combination

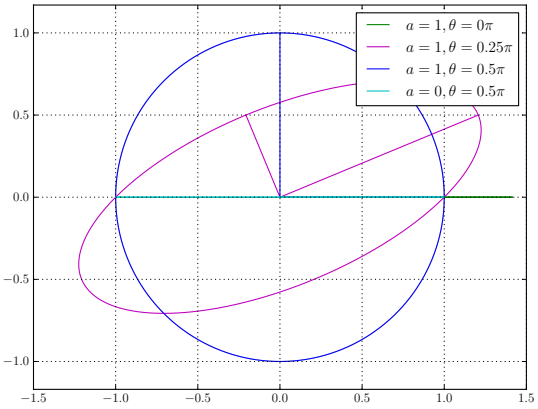
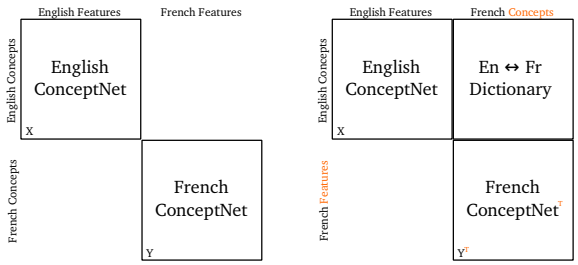


Figure 2: $K\vec{c}$ for $|\vec{c}| = 1$, representing $U_k\Sigma$, for several values of α and θ_U. x-axis represents the factor of $\vec{u}_X$; y-axis represents the factor of $\hat{a_\perp}$.



(a) No overlap = no blending

(b) Transposing one of the matrices allows for a concept-to-concept bridge

Figure 3: Bridge Blending: Connecting data that do not overlap

of the singular vectors of both X and Y. However, once $\theta_U = \pi/2$, i.e., the input vectors become orthogonal, the result is a circle, meaning that the original singular vectors passed through either unchanged or merely reordered. As the first singular vector rotates, the second also necessarily rotates in order to remain orthogonal to the first, so now both vectors have support from both X and Y. Incidentally, this means that even though the features did not overlap at first, the rotated U now projects concepts in one matrix into space that has features in both matrices. This means that by projecting a vector containing only features from X into the space and truncating it to its initial singular value, we end up with a vector that maps to features in both X and Y; this is cross-domain inference.

So we see that to maximize interaction of one pair of singular values, the relative magnitudes should be equal. Thus, (Havasi et al. 2009) used this as a heuristic for blending more complex matrices: make their first singular values equal. One possibility, as yet unexplored empirically, is to engineer interactions between multiple singular values; this can be accomplished using the "rank-1" formulation of the SVD,

$$X + Y \approx \sum_{i=0}^{k} \left(\sigma_{Xi} \vec{u_{Xi}}^T \vec{v_{Xi}} + \sigma_{Yi} \vec{u_{Yi}}^T \vec{v_{Yi}} \right),$$

by setting the σs *a priori*.

Bridge Blending

Blending only works when pairs of datasets overlap in either their rows or their columns. Consider the layout of Figure 3a,[2] for example: we have (hypothetically) commonsense knowledge in English and French, but without knowing which English concepts or features correspond to which French concepts or features, we have no way reasoning jointly over them. The similarity (dot product) between any English concept/feature and any French concept/feature in such a layout is exactly 0. In fact, it's readily shown that unless the two matrices share a singular value exactly, none of the axes will contain both English and French concepts. Rather, the set of singular vectors of the "blend" will be simply the union of the singular values of the English and French matrices alone, padded with zeros. Boring.

So how can we reason jointly over both English and French? We need to add another dataset, called a "bridge," to connect English and French. It could fill one of the missing off-diagonal entries in Figure 3, but with what? We would need data about either French features about English concepts, or English features about French concepts. We do not have that data directly, though we could possibly infer it from the English and French commonsense data. More readily available is a bilingual dictionary, connecting English concepts to French concepts and vice versa. We could transform that into a matrix of English concepts by French concepts. The bridge data could fit into the blend if we *transposed* one of the ConceptNet matrices, as in Figure 3b.

The canonical encoding of commonsense data is concepts on rows and features on columns; will the transposed arrangement still yield meaningful results? Transposing a matrix just reverses the roles of U and V in its SVD, so transposing a single matrix does no harm. But we might worry that the fact that the English and French concepts/features are on different sides of the matrix keeps them from being meaningfully related. This section gives a few steps towards a mathematical demonstration that cross-domain inference occurs in bridged blending in general and in the transposed arrangement in particular. A complete mathematical treatment awaits a future publication, but applications including ProcedureSpace (Arnold and Lieberman 2010) have empirically demonstrated the effectiveness of the technique.

Let X be the English ConceptNet with concepts x_i, and Y be the French ConceptNet with concepts y_i. We then encode the bilingual dictionary into a matrix B, where $B(i, j)$ gives the similarity between concepts x_i and y_j. We now array the two datasets X and Y along with the bridge dataset B in the transposed bridge blend layout:

[2] In practice, commonsense data matrices are about 5 times as wide as tall, since most concepts participate in several features.

$$C = \begin{bmatrix} X & B \\ & Y^T \end{bmatrix} \approx U\Sigma V^T$$

Intuitively, we suspect that the bridge dataset will cause the eigenconcepts and eigenfeatures to be composed of items from both X and Y^T. If this works, then we will be able to determine what French concepts apply to an English feature, or ask about the translation of different senses of a word based on projecting different combinations of features, all by computing matrix-by-vector products.

To see if it works, consider a simplified case that the bridge data is a weighted identity matrix, i.e., every x_i corresponds to exactly one y_j with constant weight. This setup requires that the number of rows of X equal the number of rows of Y. Though realistic bridge blends break both of these rules, this setup is still a representative idealization.

The transposed bridge layout with identity bridging is:

$$C = \begin{bmatrix} X & \alpha I \\ 0 & Y^T \end{bmatrix}.$$

If $Y^T = 0$, blending with the constant-weight identities adds α^2 to each eigenvalue without changing the eigenvectors. To analyze the contribution of Y^T, we start by computing the row-row dot products:

$$\begin{aligned} CC^T &= \begin{bmatrix} X & \alpha I \\ 0 & Y^T \end{bmatrix} \begin{bmatrix} X^T & 0 \\ \alpha I & Y \end{bmatrix} \\ &= \begin{bmatrix} XX^T + \alpha^2 I & \alpha Y \\ \alpha B & Y^T Y \end{bmatrix}. \end{aligned}$$

If $XX^T\vec{u} = \lambda\vec{u}$ (i.e., $\vec{u}$ is an eigenvector of XX^T), then

$$\begin{aligned} CC^T \begin{bmatrix} \vec{u} \\ 0 \end{bmatrix} &= \begin{bmatrix} XX^T + \alpha^2 I & \alpha Y \\ \alpha Y^T & Y^T Y \end{bmatrix} \begin{bmatrix} \vec{u} \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} XX^T\vec{u} + \alpha^2 I\vec{u} \\ \alpha Y^T\vec{u} \end{bmatrix} = \begin{bmatrix} (\lambda + \alpha^2)\vec{u} \\ \alpha Y^T\vec{u} \end{bmatrix}. \end{aligned}$$

So as long as $\vec{u}$ is not in the null space of Y^T, no vector with zero support in the Y^T domain could be an eigenvector. So the eigenconcepts of the bridge-blended data *must* be determined by both matrices. Exchanging the roles of X and Y^T, the same argument shows that eigenfeatures also must have cross-domain support.

Bridge Blending is Also CMF

Consider a general bridge blend layout:

$$\begin{bmatrix} X & Y \\ & Z \end{bmatrix} \approx \begin{bmatrix} U_{XY} \\ U_{0Z} \end{bmatrix} \begin{bmatrix} V_{X0} \\ V_{YZ} \end{bmatrix}^T = \begin{bmatrix} U_{XY}V_{X0}^T & U_{XY}V_{YZ}^T \\ U_{0Z}V_{X0}^T & U_{0Z}V_{YZ}^T \end{bmatrix}$$

The loss once again factors into separate terms for each relation, showing that bridge blending is also a kind of CMF:

$$\begin{aligned} D(\hat{A}|A) =& D(U_{XY}V_{X0}^T|X) + D(U_{XY}V_{YZ}^T|Y) + \\ & D(U_{0Z}V_{X0}^T|0) + D(U_{0Z}V_{YZ}^T|Z) \end{aligned}$$

We observe that the factor V_{YZ} ties together the factorization of X and Z through the bridge data Y; without a penalty for reconstructing Y poorly, the factorizations of X and Z would be independent. Note that one component of the loss is predicting non-zero in the zero bottom-left corner. If in fact we know that region to be non-zero, we could use a weighted loss function, which CMF permits.

Conclusion

Factored inference, as in AnalogySpace, is a useful tool for approximate reasoning over noisy knowledge bases like ConceptNet. We have shown that Blending is a kind of Collective Matrix Factorization (CMF) in that the factorization spreads out the prediction loss between each dataset. We also showed that blending additional data causes the singular vectors to rotate between vectors in different domains, which enables cross-domain inference. In a simplified example, we justified previous evidence that the maximum interaction occurs when the magnitudes (as defined by the largest singular values) of the two matrices are equal. Finally, we described and justified Bridge Blending, both normal and transposed, which is a kind of CMF that can connect more than 2 datasets.

Acknowledgements

We gratefully acknowledge helpful critique about concepts, math, and explanations from Jason Alonso, Catherine Havasi, and Rob Speer. We also thank Weike Pan of Hong Kong University of Science and Technology for bringing Collective Matrix Factorization to our attention.

References

Arnold, K. C., and Lieberman, H. 2010. Managing ambiguity in programming by finding unambiguous examples. In *Onward! 2010 (to appear)*.

Brand, M. 2006. Fast low-rank modifications of the thin singular value decomposition. *Linear Algebra and its Applications* 415(1):20 – 30. Special Issue on Large Scale Linear and Nonlinear Eigenvalue Problems.

Havasi, C.; Speer, R.; Pustejovsky, J.; and Lieberman, H. 2009. Digital Intuition: Applying common sense using dimensionality reduction. *IEEE Intelligent Systems*.

Havasi, C.; Speer, R.; and Alonso, J. 2007. ConceptNet 3: a flexible, multilingual semantic network for common sense knowledge. In *Recent Advances in Natural Language Processing*.

Havasi, C. 2009. *Discovering Semantic Relations Using Singular Value Decomposition Based Techniques*. Ph.D. Dissertation, Brandeis University.

Singh, A. P., and Gordon, G. J. 2008. Relational learning via collective matrix factorization. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 650–658. New York, NY, USA: ACM.

Speer, R. H.; Havasi, C.; Treadway, K. N.; and Lieberman, H. 2010. Finding your way in a multi-dimensional semantic space with Luminoso. In *IUI '10: Proceeding of the 14th international conference on Intelligent user interfaces*, 385–388. New York, NY, USA: ACM.

Speer, R.; Havasi, C.; and Lieberman, H. 2008. AnalogySpace: Reducing the dimensionality of common sense knowledge. *Proceedings of AAAI 2008*.