In CF-based meta learning, base models play the role of "users" $\{u\}$ while data points play the role of "items" $\{i\}$, where we borrow a convention from recommender systems by using $u$ to index into classifiers/users and $i$ to index into data/items. A classifier assigns ratings on data items in the sense that the rating tells us how likely a data point is positive in terms of a conditional probability score. To realize an effective ensemble transformation, we seek to solve the optimization problem, $argmin_{X,Y}\, J(X,Y)$, where $J(\cdot,\cdot)$ is a cost function and takes the following form:

$$J(X,Y) = \sum_{u,i} c_{ui} \cdot loss\big(r_{ui}, f(x_u, y_i)\big) + regularization \tag{1}$$

For notational convenience, let $X \in \mathbb{R}^{m\times d}$ be a matrix formed by stacking $m$ classifier vectors $\{x_u \mid u = 1 \ldots m\}$, each of which as a row vector, and let $Y \in \mathbb{R}^{n\times d}$ be the corresponding data matrix comprising $n$ data instance vectors $\{y_i \mid i = 1 \ldots n\}$. Let $r_{ui}$ denote a "rating" given by a classifier $u$ on a data item $i$, which essentially represents a conditional probability $P(y = 1|x)$ predicted by the classifier; and $x_u \in \mathbb{R}^d$ and $y_i \in \mathbb{R}^d$ represent the latent vector for the classifier and the data point, respectively, in the $d$-dimensional vector space. Further, we use $J(\cdot,\cdot)$ to denote a cost function, representing the total cost, while using $loss(\cdot,\cdot)$ to denote an individual cost incurred by a single classifier on a single data item (due to the latent-factor approximation for the ratings).

The most general form of the optimization objective is to find the latent-factor representation – i.e., appropriate configurations as $d$-dimensional vectors for $x_u$ and $y_i$ respectively – such that $f(x_u, y_i)$, a function of the latent factors, best approximates ratings $r_{ui}$ as measured by the loss function $loss(\cdot,\cdot)$, where each loss term is weighted by the corresponding confidence score $c_{ui}$. A confidence score measures the reliability of a rating: the higher the confidence score, the higher the reliability of a rating, which translates into "better" conditional probability prediction. For the positive class, we expect the rating to be high (close or equal to 1) to be confident, whereas for the negative class, we expect the rating to be low (closer or equal to 0). To work toward minimizing the (total) cost $J$ for a given confidence measure (e.g., Brier score), the loss terms associated with high confidence scores, within the summation in (1), should be kept as small as possible; that is, the latent factors $\{x_u\}$ for classifiers and $\{y_i\}$ for dataset should assume an optimal configuration such that, on average, $f(x_u, y_i)$ is pushed, as close as possible, toward each rating $r_{ui}$.

A special case is where the re-estimating function $f(x_u, y_i)$ is a simple dot product $x_u^T \cdot y_i$ and the loss function $loss(\cdot,\cdot)$ takes the form of a squared loss (between the rating $r_{ui}$ and the re-estimated value $x_u^T \cdot y_i$):

$$L(X,Y) = \sum_{u,i} c_{ui}(r_{ui} - x_u^T \cdot y_i)^2 + \lambda\left(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2\right)$$

In this case, we seek to find the inner product between $x_u$ and $y_i$ that best approximates $r_{ui}$ in a (weighted) least-squares sense, where each square-error term is weighted by its corresponding confidence score $c_{ui}$. The higher the confidence score it is assigned to a rating, the smaller the deviation from the inner product $x_u^T \cdot y_i$ to the target $r_{ui}$ it is necessary to keep the total cost $J$ small. Additionally, the $\ell 2$-regularization with parameter $\lambda$ is used to prevent overfitting during the training phase.