Chapter 5

A novel class of matching algorithms

We have answered the first two fundamental questions posed in Chapter 1. Namely, we found that there is an optimal distribution from which data should be drawn to train the learning algorithm, we called it the dual distribution and described how to find it. We then established an algorithm that can determine if a weighting scheme used to change the distribution of the training data will yield out-of-sample improvement. The remaining question is: how to find such weighting scheme? This question has received great attention in the covariate shift literature, in which various algorithms have been proposed to match the training distribution to a test distribution. The idea behind these methods is to estimate the weights $w(x) = p_S(x)/p_R(x)$, in order to correct for the covariate shift bias. In the following section, we review some of the most popular methods. We then propose a new class of algorithms that allow finding the weights efficiently and show their performance on real data sets.

5.1 Previous algorithms

The problem that instance weighting algorithms for covariate shift correction solve consists of estimating $w(x) = p_S(x)/p_R(x)$. The following methods find this ratio in different ways.

5.1.1 Indirect ratio estimation via KDE

The first approach used to find the importance weight was Kernel Density Estimation (KDE). The method finds the training and test densities by modeling each distribution as a linear combination of

kernel functions around the existing samples. That is

$$\hat{p}(x) = \frac{1}{N(2\pi\sigma^2)^{d/2}} \sum_{i=1}^{N} K_\sigma(x, x_i). \tag{5.1}$$

The method is very simple to implement, only requiring a choice for the size σ of the kernels. However, the method suffers in the case of noisy data sets as the errors in estimation get amplified when the ratio is taken between two estimated quantities.

Hence, the following methods propose estimating the ratio directly, rather than trying to first solve the hard problem of estimating full densities in order to find the ratio.

5.1.2 Logistic regression methods

The next group of methods propose to build a classifier that would discriminate data coming from the training and test distributions. The model assumes that all data is drawn from a probability distribution $p(x)$, and that there is a variable η that determines if a point comes from one or the other distribution. With this assumption and using Bayes' theorem, it is possible to estimate the ratio. Specifically, let

$$p_R(x) = p(x|\eta = -1) \quad \text{and} \quad p_S(x) = p(x|\eta = 1). \tag{5.2}$$

Then, by Bayes' theorem

$$w(x) = \frac{p_S(x)}{p_R(x)} = \frac{p(\eta = 1|x)p(\eta = -1)}{p(\eta = -1|x)p(\eta = 1)} \tag{5.3}$$

To find the quantities in the right hand side, the method assumes further that

$$p(\eta = -1)/p(\eta = 1) \approx N_R/N_S \tag{5.4}$$

and finds the remaining ratio via logistic regression. The optimization problem is given by

$$\theta^* = \arg\min_\theta \frac{1}{N_R + N_S} \sum_{x \in R \cup S} \log(1 + \exp(-\eta\theta^T x)) \tag{5.5}$$

and the final estimate for the weights is therefore

$$\hat{w}(x) = \frac{N_R}{N_S} \exp(x^T\theta^*). \tag{5.6}$$

Variations of this idea are described in [13], [14], and [71]. The problem with this approach is that the assumptions made by the model are quite strong.

5.1.3 Kernel mean matching (KMM)

One of the methods that has been used in practical applications most often is Kernel Mean Matching [38]. The success of this algorithm is based on a theorem that states that two distributions are the same, if and only if the mean in all dimensions of a Reproducing Kernel Hilbert Space (RKHS) are equal. Formally, let $\Phi : \mathcal{X} \to \mathcal{F}$ be a map into a feature space and let $\mu : \mathcal{P} \to \mathcal{F}$ denote the expectation operator,

$$\mu(P) := \mathbb{E}_{x \sim P(x)}[\Phi(x)]. \tag{5.7}$$

where $\mathcal{P}$ is a probability space. Then the theorem proved in [38] states that the operator μ is bijective if $\mathcal{F}$ is an RKHS with a universal kernel $k(x, x') = \langle \Phi(x), \Phi(x') \rangle$. In other words, for two distributions P and P', $P = P'$ iff $\mu(P) = \mu(P')$.

Hence, the authors propose to solve the following optimization problem to match the means of the samples in a RKHS:

$$\begin{aligned} \text{minimize}_w \quad & ||\mu(P_S) - \mathbb{E}_{x \sim P_R}[w(x)\phi(x)]|| \\ \text{s.t.} \quad & w(x) \geq 0,\ \mathbb{E}_{x \sim P_R(x)}[w(x)] = 1. \end{aligned} \tag{5.8}$$

In practice, the objective function of the above problem is approximated with empirical estimates:

$$\left|\left| \frac{1}{N_R} \sum_{i=1}^{N_R} w_i \phi(x_i) - \frac{1}{N_S} \sum_{j=1}^{N_S} \phi(x'_j) \right|\right| = \frac{1}{N_R^2} w^T K w - \frac{2}{N_R^2} \kappa^T w + const \tag{5.9}$$

The optimization problem becomes:

$$\begin{aligned} \text{minimize}_w \quad & \frac{1}{N_R^2} w^T K w - \frac{2}{N_R^2} \kappa^T w, \\ \text{s.t.} \quad & w \in [0, B], \quad \left| \sum_{i=1}^{N_R} w_i - N_R \right| \leq N_R \epsilon. \end{aligned} \tag{5.10}$$

Hence, the problem reduces to solving a quadratic program with inequality constraints. However, the main drawback of the method is that it is very sensitive to both the choice of the size of the kernel and the constants B and ϵ. Before Targeted Weighting was introduced, there was no method available to cross-validate different weighting schemes, and hence the choice of these parameters was problematic.

5.1.4 Parametric models for the ratio: KLIEP, LSIF, RuLSIF, etc.

Although KMM proved successful in some applications, as discussed in Chapter 4, weighting may worsen performance, and hence it was sometimes not a useful method as cross-validation was not

available (See Section 4.2). A few more methods were proposed based on the idea of estimating the ratio of the importance weight directly, by using a parametric model for the ratio. The first of these methods was Kullback-Liebler Importance Estimation Procedure (KLIEP) [67]. In this method, the ratio is modeled by the equation

$$\hat{w}(x) = \frac{p_S(x)}{p_R(x)} = \sum_{t=1}^{T} \alpha_t \phi_t(x). \tag{5.11}$$

With this model, the idea is to minimize the KL-divergence between the training and test distributions, given by

$$KL(\alpha) = \mathbb{E}_{P_S}\left[\log \frac{p_S(x)}{\hat{w}(x) p_R(x)}\right] = \mathbb{E}_{P_S}\left[\log \frac{p_S(x)}{p_R(x)}\right] - \mathbb{E}_{P_S}[\log \hat{w}(x)]. \tag{5.12}$$

Since the first term is constant with respect to the weights, the method maximizes the second term, adding the constraint that $\mathbb{E}_{x\sim P_R}[\hat{w}(x)] = \int p_S(x)dx = 1$. Using empirical estimates for the above quantities, the problem becomes:

$$\begin{aligned} \text{maximize}_\alpha \quad & \sum_{j=1}^{N_S} \log\left(\sum_t \alpha_t \phi_t(x'_j)\right) \\ \text{s.t.} \quad & \frac{1}{N_R}\sum_t \alpha_t \sum_{i=1}^{N_R} \phi_t(x_i) = 1, \quad \alpha \geq 0 \end{aligned} \tag{5.13}$$

This optimization problem is convex and the authors propose to solve it using gradient descent with constraint satisfaction at each step. The authors suggest using Gaussian kernels centered at the test points for the functions, that is $\hat{w}(x) = \sum_{j\in S} \alpha_t K_\sigma(x, x_j)$. The authors also claim that KLIEP has a model selection procedure, by cross validating the choice of kernel width, T, etc., with respect to the value of the objective function. Nevertheless, this cross-validation procedure is a measure of how good the match is between the two distributions, but is not a measure of the bottom line out-of-sample performance. As argued in Chapter 4, matching distributions using weights may actually worsen performance in some cases, depending on the tradeoff between the $MatchBenefit$ and $WeightLoss$. Targeted Weighting, on the other hand, could be used exactly for this cross-validation purpose in conjunction with KLIEP, or any of the matching methods proposed so far.

A variant of this method was proposed, in which the KL divergence was replaced by a least squares error approach. This led to least squares importance fitting (LSIF) [42]. In this method, the following

objective function is minimized:

$$\begin{aligned} J(\alpha) &:= \frac{1}{2}\mathbb{E}_{P_R}[(\hat{w}(x) - w(x))^2] \\ &= \frac{1}{2}\mathbb{E}_{P_R}[\hat{w}(x)^2] + \mathbb{E}_{P_R}[\hat{w}(x)w(x)] + \frac{1}{2}\mathbb{E}_{P_R}[w(x)^2] \\ &= \frac{1}{2}\mathbb{E}_{P_R}[\hat{w}(x)^2] + \mathbb{E}_{P_S}[\hat{w}(x)] + const. \end{aligned} \tag{5.14}$$

Once again, expectations are approximated using empirical averages, so that the optimization problem becomes:

$$\begin{aligned} \text{minimize}_\alpha \quad & \frac{1}{2}\alpha^T \hat{H}\alpha - \hat{h}\alpha \\ \text{s.t.} \quad & \alpha > 0 \end{aligned} \tag{5.15}$$

with $\hat{H}_{jk} = \frac{1}{N_R}\sum_{i=1}^{N_R}\phi_j(x_i)\phi_k(x_i)$, and $\hat{h}_i = \frac{1}{N_S}\sum_{j=1}^{N_S}\phi_i(x'_j)$. Nevertheless, the authors point out that the method is numerically unstable, so they use L2-regularization with an unconstrained problem (uLSIF). Once again, they use the objective function value in order to cross-validate the choice of kernel size

Finally, RuLSIF was proposed in which the weights are regularized, by using

$$w(x) = \frac{p_S(X)}{\beta p_R(x) + (1-\beta)p_S(x)}. \tag{5.16}$$

The need for this regularization is precisely the tradeoff pointed out in Chapter 4. Nevertheless, the authors propose finding β using importance weighted cross-validation (IWCV) [65]. However, IWCV requires knowledge of the ratio $p_S(x)/p_R(x)$. Since this ratio is unknown, cross-validation is done assuming that the estimate of the weights, found through one of the above methods, is actually accurate.

5.1.5 Discrepancy minimization

The matching algorithms discussed so far make no assumption about the learning algorithm to be used. That is, they make no assumption of what loss function is to be minimized or what the hypothesis set is. One of the most recent algorithms, introduced in [26], uses the notion of *discrepancy* [47], which leads to a tractable algorithm when we use a squared loss function, a linear model, and $\mathcal{Y} = \mathbb{R}$.

The intuition behind this approach is to find weights such that the training distribution is matched to a distribution in which the hypothesis learned will yield the same out-of-sample error as if the hypothesis had been learned with training points sampled from the test distribution. Notice that this

notion does not imply that the weights found will exactly match the training and test distributions. It will only match the out-of-sample error, which in fact is what matters to a practitioner. The hope is that by doing this, the negative effect of weighting will be minimized, as the perturbation of the weights from unity should be smaller than when we try to match exactly the two distributions.

Formally, the discrepancy is defined as follows. Let $\mathcal{H}$ be a set of functions with $h : \mathcal{X} \to \mathcal{Y}$ for every $h \in \mathcal{H}$, and let $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ define a loss function over $\mathcal{Y}$. The discrepancy distance disc_L between two distributions P and Q over $\mathcal{X}$ is given by

$$\text{disc}_L(P, Q) = \max_{h,h' \in \mathcal{H}} |L_P(h', h) - L_Q(h', h)| \tag{5.17}$$

where $L_P(h, h') = \mathbb{E}_{x \sim P}[\ell(h(x), h'(x))]$. That is, the discrepancy finds the maximum out-of-sample error difference that could result if we trained with points distributed as P instead of Q. It is not hard to see that this notion can give a bound for the error made by training with a different distribution. In fact, in [26], assuming that kernel ridge regression is used (model that uses a squared loss function, a linear model with a kernel for the non-linear transformation, ℓ_2 norm regularization for the parameters, and $\mathcal{Y} = \mathbb{R}$), the authors prove the following bound:

$$|\ell(h'(x), y) - \ell(h(x), y)| \leq 2r\sqrt{M\lambda \text{disc}(P', P)/\lambda}, \tag{5.18}$$

where $K(x, x) \leq r^2$, $L(h(x), y) \leq M$, and λ is the regularization parameter. Hence, the authors propose minimizing the discrepancy between the training and test sets. It turns out that the optimization problem is equivalent to the following semi-definite program:

$$\begin{aligned} \text{minimize}_{w,\lambda} \quad & \lambda \\ \text{s.t} \quad & \lambda I - M(w) \succeq 0 \\ & \lambda I + M(w) \succeq 0 \\ & 1^T w = 1, w \geq 0 \end{aligned} \tag{5.19}$$

where $M(w) = \sum \hat{P}_S(x) x x^T - \sum_{i=1}^{N_R} w_i x_i x_i^T$, and $A \succeq 0$ denotes matrix A is positive semi-definite. Solving this problem requires, therefore, using convex optimization packages. Since the notion of discrepancy allows regularizing the weights, the algorithm is effective. Nevertheless, its main drawback is that it is only valid for regression problems with the learning model specified above.

5.2 A new class of algorithms

We propose a new class of algorithms for matching. These algorithms were conceived for very large datasets, as it is the case with the Netflix dataset, where computational complexity is a key issue. Furthermore, as the algorithms were conceived in the recommender system setting, we wanted to be able to choose particular coordinates along which to match distributions. In recommender systems, the data usually includes only an item ID, a user ID, and the rating which is the value that the system will try to predict. The time of the rating may also be available. Under this scenario, matching training and test sets is not meaningful as both user ID and movie ID are not relevant coordinates. Rather, matching certain coordinates that can be derived from the data, such as the item popularity (known as item support), or the amount of ratings by a particular user (user support), can be desired. We begin by introducing the hard matching algorithm which is the basic building block of the algorithms presented here. A "soft" version is also presented, and finally different variants of the initial algorithm are discussed.

5.2.1 Hard matching

We recall that a sampled version of the loss function yields a variance with approximately $N_{\text{eff}} = \left(\sum_i w_i\right)^2 / \sum_i w_i^2$. Hence, as there are many schemes that can match the training dataset to a desired distribution, we pick the one that minimizes the ℓ_2-norm of w, since this will maximize N_{eff}. The reason for this is that the the numerator of the expression for N_{eff} is a constant. To see this, notice we can scale all weights by a constant and this would not have any effect on the learning algorithm. Hence we pick to normalize the sum of the weights to N_R. Here we use N_R to denote the number of points in the training set, while we use N_S for the points in the test set. To illustrate the matching condition, first consider a one-dimensional distribution. One way to match is to divide the input space into bins and match the fraction of points appearing in each of the bins for the training set and for data coming from the desired distribution.

To formalize this, let T denote the number of bins that we use to divide the input space. Let $b : \mathcal{X} \to \{1, \ldots, T\}$ be a function that indicates the bin number into which each data point x_i falls. Let $\mu, \nu \in \mathbb{R}^T$ be vectors that hold the frequency of points in each bin for the training set and for the distribution we desire to match to. That is

$$\mu_j = \frac{1}{N_R} \sum_{i=1}^{N_R} I[b(x_i) = j], \tag{5.20}$$

and similarly for ν, except that the summation is over points in the test set if $P' = P_S$. In case P' is a different distribution, such as the dual distribution, ν can be obtained by integrating the density of

P' for each of the bins. With this notation, the idea described above consists of solving the following optimization problem:

$$\begin{aligned} \text{minimize}_{w_i} \quad & \frac{1}{2}\sum_{i=1}^{N_R}(w_i - 1)^2 \\ \text{s.t.} \quad & \frac{1}{N_R}\sum_{i:b(x_i)=\tau} w_i = \nu(\tau), \quad \text{for } \tau = 1, \ldots, T \\ & w_i \geq 0, \end{aligned} \tag{5.21}$$

where $\nu(\tau)$ indicates the τ'th component of the vector ν. This is a quadratic program with linear constraints, and hence it is convex and has a unique solution. In fact we can solve for the weights analytically by constructing the Lagrangian and setting its gradient to 0. We ignore for now the positivity constraints. Notice further that since

$$\sum_{i=1}^{N_R} w_i = \sum_{\tau}\sum_{i:b(x_i)=\tau} w_i = N_R \sum_{\tau} \nu(\tau) = N_R, \tag{5.22}$$

it is not necessary to include normalization constraints for the weights as this constraint is implicitly satisfied.

The Lagrangian is given by

$$\mathcal{L}(w, \lambda) = \sum_{i=1}^{N_R}(w_i - 1)^2 + \frac{1}{N_R}\sum_{\tau=1}^{T}\lambda_\tau \sum_{i:b(x_i)=\tau} w_i - \nu(\tau), \tag{5.23}$$

thus

$$\frac{\partial \mathcal{L}(w, \lambda)}{\partial w_i} = 0 \Longrightarrow w_i = 1 - \frac{\lambda_\tau}{N_R}. \tag{5.24}$$

Notice that this means that the weights of all points that fall in the same bin must be equal. Hence, let $w_i = w_\tau$ for all $x_i \in R$ with $b(x_i) = \tau$. Then, substituting in the constraint,

$$\frac{1}{N_R}\sum_{i:b(x_i)=\tau} w_i = \frac{1}{N_R}\sum_{i:b(x_i)=\tau} N_R \mu(\tau) w_\tau = \mu(\tau) w_\tau = \nu(\tau). \tag{5.25}$$

Therefore, the solution for the weights is given by

$$w_\tau = \frac{\nu(\tau)}{\mu(\tau)}. \tag{5.26}$$

Clearly, this approach gives an approximation for the importance weighs $w(x) = p'(x)/p(x)$. Also notice that having ignored the non-negativity constraints in the problem did not change the solution,

as clearly $w_\tau \geq 0$.

Now that the intuition is clear, we move on to multiple coordinates. We want the distributions to be matched along each of the desired coordinates, and we consider projections into these coordinate independently. This allows the algorithm to grow linearly in complexity with the number of coordinates chosen, rather than exponentially in d, the number of dimensions of the input space. Generalizing the previous notation, let C be the number of coordinates we want to match along, and let $T_1, \ldots, T_C$ be the number of bins chosen along each of the coordinates. Let $b_c : \mathcal{X} \to \{1, \ldots, T_c\}$ for $c = 1, \ldots, C$ be functions that map each training point x_i to the corresponding bin number along coordinate c. Also, let $\mu_c, \nu_c \in \mathbb{R}^{T_c}$, for $c = 1, \ldots, C$ be the corresponding frequency vectors along coordinate c. The optimization problem we want to solve is given by

$$\begin{aligned} \text{minimize}_{w_i} \quad & \frac{1}{2}\sum_{i=1}^{N_R}(w_i - 1)^2 \\ \text{s.t.} \quad & \frac{1}{N_R}\sum_{i:b_c(x_i)=\tau} w_i = \nu_c(\tau), \quad \text{for} \quad \begin{matrix}\tau = 1, \ldots, T, \\ c = 1 \ldots, C\end{matrix} \\ & w_i \geq 0 \end{aligned} \tag{5.27}$$

The Lagrangian in this case is given by

$$\mathcal{L}(w, \lambda) = \frac{1}{2}\sum_{i=1}^{N_R}(w_i - 1)^2 + \frac{1}{N_R}\sum_{c=1}^{C}\sum_{\tau=1}^{T_c}\lambda_{c\tau}\sum_{i:b(x_i)=\tau} w_i - \nu_c(\tau). \tag{5.28}$$

Setting the gradient of the Lagrangian with respect to w_i to 0 yields

$$w_i = 1 - \sum_{c=1}^{C}\frac{\lambda_{cb_c(x_i)}}{N_R}. \tag{5.29}$$

We now solve for the Lagrange multipliers. We substitute in the constraint and obtain

$$\frac{1}{N_R^2}\sum_{i:b_c(x_i)=\tau}\sum_{k=1}^{C}\lambda_{kb_k(x_i)} = \mu_c(\tau) - \nu_c(\tau). \tag{5.30}$$

Notice further that for coordinate c, the above equation becomes

$$\frac{1}{N_R^2}\sum_{i:b_c(x_i)=\tau}\lambda_{cb_c(x_i)} = \mu_c(\tau) - \nu_c(\tau) - \frac{1}{N_R^2}\sum_{i:b_c(x_i)=\tau}\sum_{\substack{k=1 \\ k\neq c}}^{C}\lambda_{kb_k(x_i)}. \tag{5.31}$$

But for the left hand side, since the outer sum is over the x_i that satisfy $b_c(x_i) = \tau$, we are simply adding $\mu_c(\tau)N_R$ times the value of $\lambda_{c\tau}$. Hence,

$$\frac{\lambda_{c\tau}}{N_R} = \frac{1}{\mu_c(\tau)} \left(\mu_c(\tau) - \nu_c(\tau) - \frac{1}{N_R} \sum_{i:b_c(x_i)=\tau} \sum_{\substack{k=1 \\ k\neq c}}^{C} \frac{\lambda_{kb_k(x_i)}}{N_R} \right). \tag{5.32}$$

Rescaling the Lagrange multipliers to absorb the N_R constant, we have the following system of equations:

$$w_i = 1 - \sum_{k=1}^{C} \lambda_{kb_k(x_i)} \tag{5.33}$$

$$\lambda_{c\tau} = 1 - \frac{\nu_c(\tau)}{\mu_c(\tau)} - \frac{1}{N_R\mu_c(\tau)} \sum_{i:b_c(x_i)=\tau} \sum_{\substack{k=1 \\ k\neq c}}^{C} \lambda_{kb_k(x_i)}. \tag{5.34}$$

To solve for the Lagrange multipliers, we initialize them at 0, and use Equation 5.34 iteratively to update their values. In our experiments, the values converge in a few iterations. With these values, the weights can be readily found.

We applied this algorithm to the Netflix dataset, as this dataset suffers from the covariate shift problem. The dataset consists of around 100 million points that included the user ID, movie number, time of a rating and rating (the value to predict). That is, $R = \{(u_i, m_i, t_i, r_i)\}$. The dataset was designed in such a way that the training set included ratings from historical data, but the test set only included the most recent ratings available to Netflix. This inherently created a difference in distributions along various coordinates of the data. The coordinates included: absolute time of rating (t); time since first rating of movie (dt); number of ratings per user or 'user support' (us); number of ratings for a movie or 'movie support' (ms); time since first rating of user (ut); and order of rating among the user's ratings (un).

We ran one of the most popular algorithms used in the Netflix competition, the SVD [44], and noticed that performance actually worsened if we used hard matching. This occurred both if we matched along single coordinates, or all coordinates at once. Table 5.1 summarizes these results. This was actually not very surprising as we were aware that weights could have a negative effect on the sample 'size', and in this particular dataset, some weights needed to become very large in order to achieve the matching constraints, which worsens this effect. To alleviate this problem, we introduce a regularized version of this method. We call this method "Soft Matching", which we explain in the next subsection.

A remaining question to be answered is how to choose the free parameters T_c, the number of bins in each of the coordinates. Sometimes, the data will have a natural division for the different coordinates.

Coordinates	RMS error improvement (basis points)
t	−71
dt	−14
us	−121
ms	−147
ut	−41
un	−40
all coords.	−93

Table 5.1: RMS error improvement in basis points when the given coordinates are matched between training and test sets using the hard matching algorithm.

For example, in the Netflix dataset, movies were rated over a period of 2,243 days. Therefore, binning by day seemed a natural thing to do. Nevertheless, it is not necessarily clear from the data how many bins should be chosen. Yet, the higher the number of bins, the higher the chance that some bins will have very few points, which could lead to inaccurate estimates of $p_S(x)/p_R(x)$ for those bins. In the other extreme, if very few bins are chosen, the estimated ratio will also be inaccurate because it captures a large part of the input space. In the extreme case, the ratio will always be 1. Therefore, as a rule of thumb, in the experiments ran on UCI classification datasets and LIACC Regression datasets, we chose the number of bins to be such that on average, every bin had at least 10 points. Hence, for datasets containing in the order of 10^2 points, 10 bins were used, while for datasets in the order of 10^3, 100 bins were used. For the Netflix dataset, the bins were either: the number of days, which meant about 50,000 points per bin for the t, dt, ut, and un coordinates; the number of unique movies (17,771) for the movie support ms, which yields about 5,000 items per bin; and 3,000 bins for the user support, each one corresponding to the number of movies rated by each user, with the last bin grouping all users that rated more than 3,000 movies. This yields about 33,000 points per bin.

5.2.2 Soft Matching

In order to reduce the negative effect or $WeightLoss$ that hard matching can lead to, we alleviate the effect by softening the constraints. One way to do this is to pull the constraints into the objective function. By using free parameters that allow trading off the amount of matching desired, we can get control how close to unity we want the weights to be. We call this problem Soft Matching, which is

described as follows:

$$\begin{aligned} \text{minimize}_w \quad & \frac{1}{2}\sum_{i=1}^{N_R}(w_i-1)^2 + \frac{1}{2}\sum_{c=1}^{C}\lambda_c\sum_{\tau=1}^{T_c}\left(\sum_{i:b(x_i)=\tau}\frac{w_i}{N_R} - \nu_c(\tau)\right)^2 \\ \text{s.t.} \quad & w \geq 0. \end{aligned} \tag{5.35}$$

Notice the similarity of the above objective function with the Lagrangian of Equation 5.28. In this case, the parameters λ_c control the level of matching, rather than being Lagrange multipliers. If the λ_c are set to infinity, then the constraints must be matched exactly in order for the objective function to be finite. If on the other hand the λ_c are set 0, all weights remain equal to 1. We solve again for the weights analytically by finding the gradient with respect to the weights and setting it to 0. As it was the case for the hard matching algorithm, we initially ignore the inequality constraints. Setting the gradient of the objective function to 0 yields

$$w_i = 1 - \frac{1}{N_R}\sum_{c=1}^{C}\lambda_c\left(\sum_{j:b(x_j)=\tau}\frac{w_j}{N_R} - \nu_c(\tau)\right). \tag{5.36}$$

Define further the auxiliary variables

$$\omega_c(b_c(x_i)) = -\sum_{x_j:b_c(x_j)=b_c(x_i)}\frac{w_j}{N_R} - \nu_c(b_c(x_i)) \quad \text{for } c = 1,\ldots,C. \tag{5.37}$$

Thus,

$$w_i = 1 + \sum_{c=1}^{C}\lambda_c\omega_c(b_c(x_i)), \tag{5.38}$$

where the N_R constant has been absorbed into the λ_c's. Substituting this equation in the definition for ω_c, we can solve for this auxiliary variables:

$$\omega_c(\tau) = \frac{1}{1+\lambda_c\mu_c(\tau)}\left(\nu_c(\tau) - \mu_c(\tau) - \frac{1}{N_R}\sum_{b_c(x_i)=\tau}\sum_{\substack{k=1\\k\neq c}}^{C}\lambda_k\omega_k(b_k(x_i))\right) \tag{5.39}$$

Once again, notice the solution is parallel to the one obtained in the hard matching procedure, except that the old Lagrange multipliers, $\lambda_{c\tau}$ are now $-\omega_c(\tau)\lambda_c$. To verify that indeed in the case that λ_c goes to infinity we recover the hard matching solution, we take Equation 5.39 and multiply it

by $-\lambda_c$. We have,

$$-\lambda_c\omega_c(\tau) = \frac{-\lambda_c}{1+\lambda_c\mu_c(\tau)}\left(\nu_c(\tau) - \mu_c(\tau) - \frac{1}{N_R}\sum_{b_c(x_i)=\tau}\sum_{\substack{k=1\\k\neq c}}^{C}\lambda_k\omega_k(b_k(x_i))\right)$$
$$= \frac{1}{1/\lambda_c+\mu_c}\left(\mu_c(\tau) - \nu_c(\tau) - \frac{1}{N_R}\sum_{b_c(x_i)=\tau}\sum_{\substack{k=1\\k\neq c}}^{C}-\lambda_k\omega_k(b_k(x_i))\right). \tag{5.40}$$

Now, taking the limit of the equation as $\lambda_c \to \infty$ we obtain

$$\lambda_{c\tau} = 1 - \frac{\nu_c(\tau)}{\mu_c(\tau)} - \frac{1}{N_R\mu_c(\tau)}\sum_{b_c(x_i)=\tau}\sum_{\substack{k=1\\k\neq c}}^{C}\lambda_{k\tau}, \tag{5.41}$$

where we have set

$$\lim_{\lambda_c\to\infty} -\lambda_c\omega_c(\tau) := \lambda_{c\tau}. \tag{5.42}$$

As it is clear, we have recovered the Hard Matching systems of equations (Eq. 5.33).

We now apply the Soft Matching algorithm to the Netflix dataset and obtain the results shown in Table 5.2. As it is clear from the table, choosing certain coordinates, and for low values of λ, there is an improvement in the RMS error on the test data. This had not occurred when we applied the hard matching algorithm. It is important to note that in this dataset, it is extremely hard to obtain improvements over available solutions, as practitioners tried for two years to improve upon solutions in order to win the Netflix Prize competition. An improvement of a few basis points, as the ones shown in Table 5.2, could have meant being ahead of the pack by a significant amount.

Picking the value for the λ_c can be done through cross-validation, which can be achieved using the Targeted Weighting algorithm introduced in Chapter 4. Once we observed which values of λ_c worked well for each of the coordinates, we matched simultaneously the subset of coordinates that TW indicated were helpful. We then tried different values of λ_c and matched simultaneously the favorable coordinates, namely the user support us, the movie support ms, and the absolute time t.

Having successfully applied Soft Matching to a real dataset that suffered from covariate shift, we explore alternative formulations of the hard matching problem, on the one hand with the idea of reducing the number of free parameters to be adjusted via cross-validation, and on the other, expressing the "softening" of the initial algorithm in a principled way. We discuss these variants in the following subsection.

Table 5.2: Soft Matching applied to the Netflix dataset, with the SVD++ learning model using 50 factors. RMS improvement is given in basis points.

Coord.	λ	RMS improvement	λ	RMS improvement
t		−71		**9**
dt		−14		**0.1**
us	1000	−121	100	**5**
ms		−147		−5
ut		−41		−20
un		−40		−17
t		**7**		**2**
dt		**0.1**		**0.1**
us	50	**9**	10	**3**
ms		−0.3		**0.4**
ut		−14		−5
un		−13		−5
t		**1**		**0.3**
dt		**0.1**		**0.01**
us	5	**2**	1	**0.4**
ms		**0.2**		**0.01**
ut		−2		−0.01
un		−2		−0.01
t, us, ms	100, 100, 10	**10**	100, 50, 10	**15**

5.2.3 Hard matching with slack variables

The second variation of hard matching we consider consists of using slack variables for the constraints, rather than including a weighted version of the constraints in the objective function. This approach accounts for the fact that even if two samples come from the same distribution, their realizations will yield slightly different histograms along the desired coordinates. For this and the following problems we state the optimization problem considering $C = 1$. Natural extensions for $C > 1$ only involve summations over the number of coordinates. The optimization problem is:

$$\text{minimize}_{w_i} \quad \frac{1}{2}\sum_{i=1}^{N_R}(w_i - 1)^2 \tag{5.43}$$

$$\text{s.t} \quad \sum_{x_i : b(x_i)=\tau} \frac{w_i}{N_R} - \nu(\tau) = \xi_\tau \quad \text{for } \tau = 1, \ldots, T$$

$$\frac{1}{2} \quad \sum_{\tau=0}^{T} \xi_\tau^2 \leq K \tag{5.44}$$

The Lagrangian of this problem is given by

$$\mathcal{L}(w,\lambda,\alpha) = \frac{1}{2}\sum_{i=1}^{N_R}(w_i-1)^2 + \sum_{\tau=1}^{T}\lambda_\tau \sum_{x_i:b(x_i)=\tau}\frac{w_i}{N_R} - \nu(\tau) - \xi_\tau + \alpha\left(\frac{1}{2}\sum_{\tau=0}^{T}\xi_\tau^2 - K\right) \tag{5.45}$$

The Karush-Kuhn-Tucker (KKT) conditions of this problem yield

$$w_i = 1 - \frac{\lambda_\tau}{N_R} \tag{5.46}$$

$$\lambda_\tau = \alpha N_R \epsilon_\tau \tag{5.47}$$

$$0 = \alpha\left(\frac{1}{2}\sum_{\tau=1}^{T}\epsilon_\tau^2 - K\right) \tag{5.48}$$

$$\alpha \geq 0. \tag{5.49}$$

The KKT system also includes the constraints. Here, Equations 5.46 and 5.47 are the result of taking the partial derivative with respect to the weights and to the Lagrange multipliers of the Lagrangian. The remaining two are the complementary slackness conditions and the positivity constraint of Lagrange multipliers for inequality constraints. We eliminate λ_τ by combining Equations 5.46 and 5.47, and then, substituting in the constraint obtain

$$w_i = \frac{\nu(\tau) + \frac{1}{\alpha}}{\mu(\tau) + \frac{1}{\alpha}}, \tag{5.50}$$

where $\tau = b(x_i)$. Notice that we arrive at the same solution we had for the Soft Matching algorithm, except that now the regularization parameters, λ, are replaced by the Lagrange multiplier α. In this problem however, α is found through a different method, by using the complementary slackness condition given in Equation 5.48. Notice that if $\alpha = 0$, then the constraints are trivially satisfied. Therefore, for complementary slackness to hold, it is necessary that

$$\frac{1}{2}\sum_{\tau=1}^{T}\epsilon_\tau^2 = K. \tag{5.51}$$

Expressing this equation in terms of α, we obtain

$$\frac{1}{2}\sum_{\tau=1}^{T}\left(\frac{\nu(\tau)-\mu(\tau)}{1+\alpha\mu(\tau)}\right)^2 - K = 0. \tag{5.52}$$

We can solve for α in the above equation numerically, for example, using the bisection method. Notice that for large enough α the first term tends to 0, so that the left hand side is a negative number. On

the other hand, for $\alpha = 0$ the first term is larger than K, otherwise the constraints would already be satisfied. Hence, the left hand side, as a function of α, goes from negative to positive in the interval $[0, \infty)$. Thus, a solution can be found through numerical methods.

This formulation trades off setting the regularization parameter λ, for the parameter K. Once again K must be determined. Yet, perhaps more information is available to find this K. For example, one can generate two samples from an estimate of the distribution P_S, and compute the expected value of K through Monte Carlo simulations. Hence, this approach gives the advantage of determining the free parameters through a method different than cross-validation.

The drawback, however, is that when multiple coordinates are considered, once again we require solving iteratively an equation for the α_c's. However, this time each equation is solved through a numerical method like the bisection method, rather than by simple substitution of values as in the Soft Matching case. This slows down considerably the algorithm. Since the solutions given by this algorithm and Soft Matching are practically the same, it is not surprising that the out-of-sample improvement is the same as the one obtained when using Soft Matching. Hence, we prefer to use Soft Matching due to the computational advantage.

5.2.4 Statistical approach

An alternative formulation similar to the slack variable approach, is to use the conditions from statistics. The Kolmogorov-Smirnov test that determines if two samples come from the same distribution can be used to specify the matching condition that two samples must satisfy. This procedure tests the null hypothesis H_0, that states that two samples come from the same distribution. Let $F_m(x)$ and $G_m(x)$ be the empirical cdf's, of the two samples with m and n points, respectively. The test accepts the null hypothesis H_0 if

$$\sqrt{\frac{mn}{m+n}} \sup_x (F_m(x) - G_n(x)) \leq D_{mn}, \tag{5.53}$$

where D_{mn} is a value that depends on the statistical significance of the test. Hence, we can set up the following optimization problem:

$$\text{minimize}_{w_i} \quad \frac{1}{2} \sum_i (w_i - 1)^2 \tag{5.54}$$

$$\text{s.t.} \quad \left| \sum_{\tau=1}^{t} \frac{w_i}{N_R} - \nu(\tau) \right| \leq D_{N_R N_S}, \quad \text{for } t = 1, \ldots, T, \tag{5.55}$$

where $D_{N_R N_S}$ is given by Kolmogorov-Smirnoff tables. Hence, in the limit where T is $\max(N_R, N_S)$, this is equivalent to carrying out the Kolmogorov-Smirnoff test. We rewrite the problem to have twice

the constraints, eliminating the absolute value, and it becomes

$$\begin{aligned} \text{minimize}_{w_i} \quad & \frac{1}{2}\sum_i (w_i - 1)^2 \\ \text{s.t.} \quad & \sum_{\tau=1}^{t} \frac{w_i}{N_R} - \nu(\tau) \leq D_{N_R N_S}, \quad \text{for } \tau = t, \ldots, T, \\ & \sum_{\tau=1}^{t} \nu(\tau) - \frac{w_i}{N_R} \leq D_{N_R N_S}, \quad \text{for } \tau = t, \ldots, T, \end{aligned} \tag{5.56}$$

The Lagrangian of this problem is given by

$$\begin{aligned} \mathcal{L}(w, \alpha, \beta) = \frac{1}{2}\sum_i (w_i - 1)^2 + \sum_{t=1}^{T} \alpha_t \left(\sum_{\tau=1}^{t} \left(\sum_{i:b(i)=\tau} \frac{w_i}{N_R} - \nu(\tau) \right) - D_{N_R N_S} \right) + \\ \sum_{t=1}^{T} \beta_t \left(\sum_{\tau=1}^{t} \left(\sum_{\tau=1}^{T} \nu(\tau) - \sum_{i:b(i)=\tau} \frac{w_i}{N_R} \right) - D_{N_R N_S} \right). \end{aligned} \tag{5.57}$$

The KKT conditions yield

$$w_i = 1 + \sum_{\tau=t}^{T} \frac{\alpha_\tau - \beta_\tau}{N_R} \tag{5.58}$$

$$0 = \alpha_t \sum_{\tau=1}^{t} \left(\left(\sum_{i:b(i)=\tau} \frac{w_i}{N_R} - \nu(\tau) \right) - D_{N_R N_S} \right) \quad \text{for } t = 1, \ldots, T \tag{5.59}$$

$$0 = \beta_t \sum_{\tau=1}^{t} \left(\left(\nu(\tau) - \sum_{i:b(i)=\tau} \frac{w_i}{N_R} \right) - D_{N_R N_S} \right) \quad \text{for } t = 1, \ldots, T \tag{5.60}$$

$$\alpha_\tau \geq 0 \tag{5.61}$$

$$\beta_\tau \geq 0, \tag{5.62}$$

where $t = b(x_i)$, and the system is completed by the constraints. We are now interested in finding α_τ and β_τ in order to determine the weights. We substitute the value of the weights given by Equation 5.58 into Equation 5.59. We obtain that either $\alpha_k = 0$ or

$$\sum_{t=1}^{k} \sum_{\tau=t}^{T} \frac{\alpha_\tau - \beta_\tau}{N_R} = \sum_{\tau=1}^{k} \mu(\tau) - \nu(\tau) - D_{N_R N_S} \quad \text{for } k = 1, \ldots, T. \tag{5.63}$$

Hence, for $k = 1$

$$\sum_{\tau=1}^{T} \frac{\alpha_\tau - \beta_\tau}{N_R} = \frac{\mu(1) - \nu(1) - D_{N_R N_S}}{\mu_1}. \tag{5.64}$$

Similarly, using Equation 5.60, we obtain that either $\beta_k = 0$ or

$$\sum_{t=1}^{k}\sum_{\tau=t}^{T}\frac{\alpha_\tau - \beta_\tau}{N_R} = D_{N_R N_S} - \sum_{\tau=1}^{k}\mu(\tau) - \nu(\tau) \quad \text{for } k = 1, \ldots, T. \tag{5.65}$$

Evaluating for $t = 1$ yields

$$\sum_{\tau=1}^{T}\frac{\alpha_\tau - \beta_\tau}{N_R} = \frac{D_{N_R N_S} - (\mu(1) - \nu(1))}{\mu_1}. \tag{5.66}$$

Since we obtain for each inequality a value for $\sum_{\tau=1}^{T}\frac{\alpha_\tau - \beta_\tau}{N_R}$ with different sign, we simply check which constraint is being violated. Violation of one constraint will imply α_1 or β_1 equal to 0, and one of the above equations is not valid. Hence, we can decide what the correct value is for $\sum_{\tau=1}^{T}\frac{\alpha_\tau - \beta_\tau}{N_R}$. Once this value is known, we can uniquely determine w_1. Subsequently, w_2 can be obtained, and so on until we have w_T.

Hence, this is another method that can be used to carry out matching. The free parameter now, rather than the regularization parameters λ as in Soft Matching, or the value of K as in hard matching with slack variables, is the value of $D_{N_R N_S}$, which must be determined using Kolmogorov-Smirnoff test values. The drawback of this method is once again the extension to multiple coordinates. In this case, the system of equations given by the KKT conditions cannot be solved in such a straightforward way. Finally, we present a different approach based on a probabilistic assumption.

5.2.5 Probabilistic approach

The final variation we present uses a similar idea as the previous methods, except that it uses a probabilistic formulation. Letting ν' and ν'' represent the frequency vectors of two samples, with each component corresponding to the frequency count in each bin, we think of these as two realizations of points sampled from a distribution P. Let N' and N'' be the number of points in each sample. We ask the question, how different can these two vectors be, given that they were generated according to the same distribution. This condition will become our new matching criterion.

Specifically, we are concerned with the quantity

$$\begin{aligned} \text{maximize}_p \quad & \frac{Pr(\mu, \nu|p)}{\mathbb{E}_{\rho', \rho''}[Pr(\rho', \rho'')|p} \\ & \sum_{i=1}^{T} p_i = 1 \end{aligned} \tag{5.67}$$

where $Pr(X)$ denotes the probability of event X occurring. In this case, we are concerned about

finding the probability that μ and ν were samples generated from p, a quantity that we normalize by its expected value.

We first simplify this expression. Notice that

$$P(\nu'|p) = \frac{N!}{(N'p_1)!\cdots(N'p_T)!} p_1^{N'p_1}\cdots p_T^{N'p_T}. \tag{5.68}$$

Further, recall Stirling's approximation, namely

$$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n = e^{n\log n - n - \frac{1}{2}\log(2\pi n)}. \tag{5.69}$$

Substituting and canceling out terms, we obtain

$$P(\nu'|p) \approx \frac{\sqrt{2\pi N'}}{\prod_{i=1}^T \sqrt{2\pi N' p_t}} \tag{5.70}$$

We now make the following simplifying assumption

$$\mathbb{E}_\rho[\rho|p] = \left(\frac{2\pi N}{\prod_{t=1}^T 2\pi N p_t}\right)^\gamma \tag{5.71}$$

Rewriting Equation 5.67, the problem is:

$$\text{maximize}_p \frac{\left(\frac{N'!\prod_{i=1}^T p_i^{N'\nu_i'}}{(N'\nu_1')!\cdots(N'\nu_T')!}\right)\left(\frac{N''!\prod_{i=1}^T p_i^{N''\nu_i''}}{(N''\nu_1'')!\cdots(N''\nu_T')!}\right)}{\left(\frac{2\pi N'}{\prod_{i=1}^T 2\pi N' p_i}\right)^\gamma \left(\frac{2\pi N''}{\prod_{i=1}^T 2\pi N'' p_i}\right)^\gamma} \tag{5.72}$$

Grouping all constant terms into the constant K, the objective function is

$$\text{maximize}_p\, K \prod_{i=1}^T p_i^{N'\nu' + N''\nu'' + 2\gamma}. \tag{5.73}$$

Writing the Lagrangian for this problem, setting its gradient to 0, and solving for p, we obtain

$$p_i^* = \frac{N'\nu_i' + N''\nu_i'' + 2\lambda}{N' + N'' + 2\gamma T}. \tag{5.74}$$

With this value of p^*, we can now compute the maximum value of the objective function. This will in turn be a criterion for determining if two samples came indeed from the same distribution. We

denote this value L_{max}, and it is given by

$$L_{\max} = \frac{N'!N''!(2\pi N')^{\gamma(T-1)}(2\pi N'')^{\gamma(T-1)}}{\prod_{i=1}^{T}(N'\nu_i')!(N''\nu_i i'')!}\prod_{i=1}^{T}\left(\frac{N'\nu_i' + N''\nu_i'' + 2\gamma}{N' + N'' + 2\gamma T}\right)^{N'\nu_i' + N''\nu_i'' + 2\gamma}. \tag{5.75}$$

Now, this probabilistic approach to the problem reduces to finding weights such that the weighted sample and the sample from the desired distribution have a normalized probability of coming from the same distribution above certain threshold. This threshold can be set to a fraction of $L_{\max}$. That is, the optimization problem becomes

$$\min_{w} \quad \frac{1}{2}\sum_{i}(w_i - 1)^2 \tag{5.76}$$

$$\text{s.t.} \quad \frac{N_R!N'!(4\pi^2 N_R N')^{\gamma(T-1)}}{\prod_{i=1}^{T}\left(\sum_{j:b(x_j)=i} w_j\right)!(N'\nu(i)')!}\prod_{i=1}^{T}\left(\frac{\sum_{j:b(x_j)=i} w_j + N'\nu(i)' + 2\gamma}{N' + N'' + 2\gamma T}\right)^{\sum_{j:b(x_j)=i} w_j + N'\nu(i)' + 2\gamma} \geq L \tag{5.77}$$

We can take the logarithm of the constraint, and we realize that the constraint is neither convex nor concave, as it involves sums of entropies and negative entropies. To see this, ignoring terms that do not involve w, and setting $\gamma = 0.5$, we obtain

$$\sum_{\tau=1}^{T}\left(\sum_{i:b(i)=\tau} w_i + N'\nu(\tau)' + 1\right)\log\left(\frac{\sum_{i:b(i)=\tau} w_i + N'\nu(\tau)' + 1}{N_R + N_S + T}\right) - \sum_{i:b(x_i)=\tau} w_i \log\left(\frac{\sum_{i:b(i)=\tau} w_i}{N_R}\right) \geq \ell_0. \tag{5.78}$$

Hence, although we begin with a more principled criterion to determine the extent of matching, the resulting optimization problem is harder to solve. Particularly, the extension to multiple coordinates is not as straightforward as for the previous problems. Secondly, due to the non-convexity of the problem, we cannot guarantee that once we find a minimum, it is the global one. Finally, we see that once again the free parameter ℓ_0 has to be chosen, although the order of it can be determined by computing $\log(L_{\max})$, if we use an estimate for p.

This concludes our exploration of matching methods, answering the third fundamental question we posed in Chapter 1. In practice we use Soft Matching as it not only gives tangible improvements in real datasets, but it is also the least demanding in terms of computation. Since we were precisely interested in finding a suitable method for large datasets, this gives Soft Matching the edge over these algorithms.