

Pre-Release Prediction of Crowd Opinion on Movies by Label Distribution Learning

Xin Geng* and Peng Hou

School of Computer Science and Engineering
Southeast University, Nanjing, China
{xgeng, hpeng}@seu.edu.cn

Abstract

This paper studies an interesting problem: is it possible to predict the crowd opinion about a movie before the movie is actually released? The crowd opinion is here expressed by the distribution of ratings given by a sufficient amount of people. Consequently, the pre-release crowd opinion prediction can be regarded as a Label Distribution Learning (LDL) problem. In order to solve this problem, a Label Distribution Support Vector Regressor (LDSVR) is proposed in this paper. The basic idea of LDSVR is to fit a sigmoid function to each component of the label distribution simultaneously by a multi-output support vector machine. Experimental results show that LDSVR can accurately predict peoples's rating distribution about a movie just based on the pre-release metadata of the movie.

1 Introduction

The movie industry is a worldwide business worth tens of billions of dollars. Thousands of new movies are produced and shown in movie theatres each year, among which some are successful, many are not. For movie producers, the increasing cost and competition boosts the investment risk. For movie audience, the prevalent immodest advertisement and promotion makes it hard to choose a movie worth watching. Therefore, both sides demand a reliable prediction of what people will think about a particular movie before it is actually released or even during its planing phase. However, to our best knowledge, there is little work, up to the present, on pre-release prediction of the crowd opinion about movies.


Aside from the unstructured reviews and discussions about the movie [Diao *et al.*, 2014], an explicit and well-structured reflection of the crowd opinion might be the distribution of ratings given by the audience who have watched the movie, just as many movie review web sites collect from their users, such as IMDb and Netflix. Note that the average rating is not a good indicator of the crowd opinion because the averaging process mingles those who like the movie and those

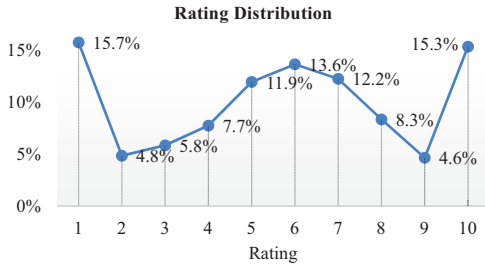
who dislike it. From the marketing point of view, a movie with controversial crowd opinions (i.e., the rating distribution concentrates at both low and high ratings) is generally much more successful than another movie with a consistent medium rating. But they might have similar average ratings. Fig. 1 gives a typical example of such case. According to the data from IMDb, the movies *Twilight* and *I, Frankenstein* both have the same average rating 5.2/10. But the top two popular ratings in the rating distribution of *Twilight* are the lowest rating 1 (15.7%) and the highest rating 10 (15.3%), respectively, while those of *I, Frankenstein* concentrate at the medium ratings 6 (21.4%) and 5 (20.1%). As a result, the budget/gross ratio of *Twilight* is \$37m/\$191m and that of *I, Frankenstein* is \$65m/\$19m. Obviously, the former movie is more worthy to invest and watch. Note that the usage of the rating distribution is not limited to gross prediction, but includes marketing strategy, advertising design, movie recommendation, etc.

It is worth emphasizing, as will be further discussed in Section 2, that predicting the crowd opinion (overall rating distribution) is quite different from predicting the individual opinion (person-specific rating). The latter problem has been extensively studied in the area of recommender systems [Adomavicius and Tuzhilin, 2005]. While personalized rating is valuable when recommending a movie to a particular user, it does not mean much when analysing the crowd opinion toward a movie. Moreover, recommendation according to the individual rating prediction is worthwhile even after many users have already watched the movie, as long as the target user has not. But crowd opinion prediction is generally only useful before any user ratings are available. As a result, the prediction should only be based on the metadata available before the movie is released.


Instead of putting the pre-release crowd opinion prediction in the recommender system scenario, this paper regards it as a *Label Distribution Learning* (LDL) [Geng *et al.*, 2013; 2010] problem since the rating distribution can be naturally viewed as a label distribution for each movie. According to the characteristics of the movie rating distribution, we propose a novel *Label Distribution Support Vector Regressor* (LDSVR), which can give multivariate and probabilistic output. The key idea of LDSVR is to fit a sigmoid function to each component of the distribution simultaneously by a support vector machine.

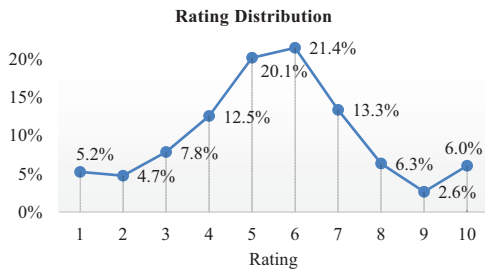
*This research was supported by NSFC (61273300, 61232007), JiangsuSF (BK20140022), and the Key Lab of Computer Network and Information Integration of Ministry of Education of China.

	Title	Twilight
	Average Rating	5.2/10
	Budget	\$ 37 Million
	Gross	\$ 191 Million



(a)

	Title	I, Frankenstein
	Average Rating	5.2/10
	Budget	\$ 65 Million
	Gross	\$ 19 Million



(b)

Figure 1: Even with the same average rating, different rating distributions (crowd opinions) of (a) *Twilight* and (b) *I, Frankenstein* result in different marketing performance.

The rest of this paper is organized as follows. Section 2 introduces some existing work related to pre-release crowd opinion prediction. After that, the proposed method LDSVR is described in Section 3. Then, experimental results are reported in Section 4. Finally, conclusions and discussions are given in Section 5.

2 Related Work

Movie rating prediction (abbreviated as RP) is a widely investigated topic in the context of recommender systems [Adomavicius and Tuzhilin, 2005], where the problem of movie recommendation is reduced to predicting movie ratings for a particular user and then recommend the movie with the highest rating to the user. The RP methods in such case are usually classified into three categories [Marovic *et al.*, 2011], i.e., content-based methods [Soares and Viana, 2014; Pilászy and Tikk, 2009], collaborative methods [Bresler *et al.*, 2014; Diao *et al.*, 2014], and hybrid methods [Amolochitis *et al.*, 2014; Jin *et al.*, 2005]. However, the pre-release crowd opinion prediction (abbreviated as PCOP) problem raised in this paper is fundamentally different from the aforementioned RP problem mainly in the following three aspects.

1. RP usually aims to predict the rating for a particular user while PCOP aims to predict the overall rating distribution generated from many users' ratings, i.e., the crowd opinion rather than the individual opinion.
2. Existing RP methods usually require previous ratings on the already watched movies from the target user (content-based methods) or other users with similar preferences (collaborative methods). On the other hand, PCOP methods do not require any previous rating data once the model is trained because they do not need to learn the preference of any particular user.
3. Most RP methods, especially the prevailing collaborative methods, cannot predict the rating of a movie until it is officially released. Many of them further rely on a sufficient number of users who have already watched and rated that movie. However, PCOP methods can make predictions before the movie is actually released or even as early as during its planning phase.

Another more related work is *Label Distribution Learning* (LDL) [Geng and Ji, 2013] recently proposed to deal with a new machine learning paradigm where each instance is annotated by a label distribution rather than a single label (single-label learning) or multiple labels (multi-label learning). The label distribution covers a certain number of labels, representing the degree to which each label describes the instance. Let $\mathcal{X} = \mathbb{R}^q$ denote the input space, $\mathcal{Y} = \{y_1, y_2, \dots, y_c\}$ denote the complete set of labels, and d_x^y denote the description degree of the label $y \in \mathcal{Y}$ to the instance $x \in \mathcal{X}$. Given a training set $S = \{(x_1, d_1), (x_2, d_2), \dots, (x_n, d_n)\}$, where $d_i = [d_{x_i}^{y_1}, d_{x_i}^{y_2}, \dots, d_{x_i}^{y_c}]^T$ is the label distribution associated with the instance x_i , the goal of LDL is to learn a mapping from $x \in \mathbb{R}^q$ to $d \in \mathbb{R}^c$ based on S . Geng *et al.* [2013] construct the mapping via a conditional mass function $p(y|x)$ formulated as a maximum entropy model. Then they use Improved Iterative Scaling (IIS) [Pietra *et al.*, 1997] or BFGS [Nocedal and Wright, 2006] to minimize the Kullback-Leibler divergence between the predicted distribution and the ground truth distribution, resulting in two LDL algorithms named IIS-LLD and BFGS-LLD, respectively. They also propose a neural-network-based approach named CPNN when the labels can be ordered [Geng *et al.*, 2013]. As a learning framework more general than single-label and multi-label learning, LDL has been successfully applied to various problems, such as facial age estimation [Geng *et al.*, 2013; 2010], head pose estimation [Geng and Xia, 2014], and multi-label ranking for natural scene images [Geng and Luo, 2014]. If we regard the movie pre-release feature vector as the instance x , the rating as the label y and the rating distribution as the label distribution d , then the PCOP problem can be naturally viewed as an LDL problem.

3 Label Distribution Support Vector Regression

In this section, we propose a *Label Distribution Support Vector Regressor* (LDSVR) for pre-release crowd opinion prediction. Compared with standard SVR [Drucker *et al.*, 1996], LDSVR must address two additional issues: (1) How to out-

put a distribution composed by multiple components? (2) How to constrain each component of the distribution within the range of a probability, i.e., $[0, 1]$? The first issue might be tackled by building a single-output SVR for each component respectively. But as pointed out in [Pérez-Cruz *et al.*, 2002], this will cause the problem that some examples beyond the insensitive zone might be penalized more than once. It is also prohibitive to view the distribution as a structure and solve the problem via structured prediction [Tsochantaridis *et al.*, 2004] because there is no direct way to define the auxiliary discriminant function that measures the compatibility between a movie and its rating distribution. A more rational solution to this issue might root in the *Multivariate Support Vector Regression* (M-SVR) [Fernández *et al.*, 2004], which can output multiple variables simultaneously. For the second issue, we are inspired by the common practice in classification tasks to fit a sigmoid function after the SVM when a probabilistic output for each class is expected [Platt, 1999]. For regression tasks, the sigmoid function could directly act as the target model of the regression instead of a post-process. Thus, the basic idea of LDSVR is, in short, to *fit a sigmoid function to each component of the label distribution simultaneously by a support vector machine*.

Suppose the label distribution \mathbf{d} of the instance \mathbf{x} is modeled by an element-wise sigmoid vector

$$\begin{aligned} \mathbf{d} &= f(\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{W}\varphi(\mathbf{x}) - \mathbf{b})} \\ &= g(\mathbf{z}) = \frac{1}{1 + \exp(-\mathbf{z})}, \end{aligned} \quad (1)$$

where $\varphi(\mathbf{x})$ is a nonlinear transformation of \mathbf{x} to a higher-dimensional feature space $\mathbb{R}^{\mathcal{H}}$, $\mathbf{W} \in \mathbb{R}^{c \times \mathcal{H}}$ and $\mathbf{b} \in \mathbb{R}^c$ are the model parameters, $\mathbf{z} = \mathbf{W}\varphi(\mathbf{x}) + \mathbf{b}$, and $g(\mathbf{z})$ is a short representation for the vector obtained by applying the sigmoid function $g(\cdot)$ to each element of \mathbf{z} . Then, we can generalize the single-output SVR by minimizing the sum of the target functions on all dimensions

$$\Gamma(\mathbf{W}, \mathbf{b}) = \frac{1}{2} \sum_{j=1}^c \|\mathbf{w}^j\|^2 + C \sum_{i=1}^n L(u_i), \quad (2)$$

where \mathbf{w}^j is the transpose of the j -th row of \mathbf{W} and $L(u_i)$ is the loss function for the i -th example. In standard SVR [Drucker *et al.*, 1996], the unidimensional loss is defined as a hinge loss function

$$L_h(u_i^j) = \begin{cases} 0 & u_i^j < \varepsilon, \\ u_i^j - \varepsilon & u_i^j \geq \varepsilon, \end{cases} \quad (3)$$

$$u_i^j = |d_i^j - f^j(\mathbf{x}_i)|, \quad (4)$$

where d_i^j and $f^j(\mathbf{x}_i)$ are the j -th elements in the corresponding vectors. This will create an insensitive zone determined by ε around the estimate, i.e., the loss less than ε will be ignored. If we directly sum the loss functions on all dimensions, i.e., $L(u_i) = \sum_j L_h(u_i^j)$, then, as pointed out in [Pérez-Cruz *et al.*, 2002], some examples beyond the insensitive zone might be penalized more than once. Fig. 2 illustrates

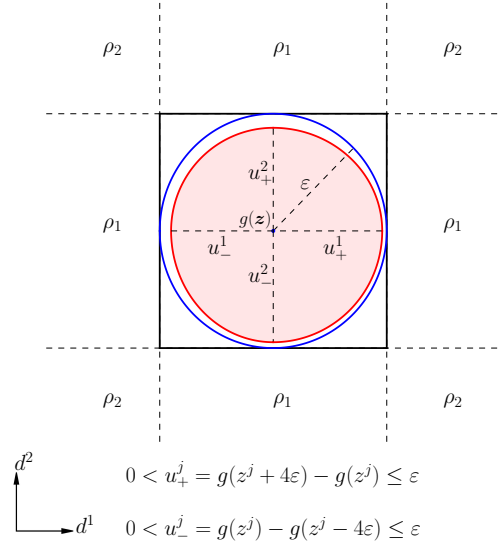


Figure 2: The insensitive zones around the estimate $f(\mathbf{x}) = g(\mathbf{z})$ in the bivariate $(d^1$ and $d^2)$ output space. The black square represents the hyper-cubic insensitive zone for the two single-output SVRs. The blue circle represents the hyper-spherical insensitive zone for M-SVR. The shaded area represents the insensitive zone for LDSVR.

this problem via a bivariate regression case, where the black square represents the hyper-cubic insensitive zone for the two single-output SVRs. As can be seen, all the examples falling into the area ρ_1 will be penalized once while those falling into the area ρ_2 will be penalized twice. Also, the L_1 -norm u_i^j in Eq. (3) is calculated dimension by dimension, which makes the solution complexity grow linearly with the increase of dimensionality [Fernández *et al.*, 2004]. As suggested in [Fernández *et al.*, 2004], we can instead use the L_2 -norm to define $L(u_i)$ so that all dimensions can join the same constraint and yield the same support vector, i.e.,

$$L(u_i) = \begin{cases} 0 & u_i < \varepsilon, \\ (u_i - \varepsilon)^2 & u_i \geq \varepsilon, \end{cases} \quad (5)$$

$$u_i = \|\mathbf{e}_i\| = \sqrt{\mathbf{e}_i^T \mathbf{e}_i}, \quad (6)$$

$$\mathbf{e}_i = \mathbf{d}_i - f(\mathbf{x}_i). \quad (7)$$

This will generate a hyper-spherical insensitive zone with the radius ε , which is represented by the blue circle in Fig. 2. Unfortunately, substituting Eq. (1), (5)-(7) into Eq. (2) does not lead to a convex quadratic form and the optimization process will not just depend on inner product. Therefore, it is hard to find the optimum as well as to apply the kernel trick.

To solve this problem, we propose an alternative loss function which can reform the minimization of Eq. (2) to a convex quadratic programming process depending only on inner product. Note that Eq. (6) calculates the Euclidean distance from the estimate $f(\mathbf{x}_i) = g(\mathbf{z}_i)$ to the ground truth \mathbf{d}_i . We can instead measure the loss by calculating how far away from \mathbf{z}_i another point $\mathbf{z}'_i \in \mathbb{R}^c$ should move to get the same output with the ground truth, i.e., $g(\mathbf{z}'_i) = \mathbf{d}_i$. Solving this

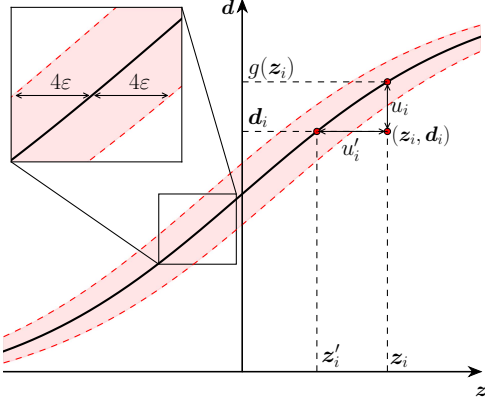


Figure 3: The relationship between u_i and u'_i . The boundaries of the insensitive zone defined on u'_i have equal distance to the sigmoid curve horizontally, but not vertically.

equation yields $z'_i = -\log(1/d_i - 1)$. Thus, the distance u'_i from z'_i to z_i can be calculated by

$$u'_i = \|e'_i\| = \sqrt{(e'_i)^T e'_i}, \quad (8)$$

$$e'_i = z'_i - z_i = -\log\left(\frac{1}{d_i} - 1\right) - (\mathbf{W}\varphi(\mathbf{x}_i) + \mathbf{b}). \quad (9)$$

The relationship between u_i and u'_i is illustrated in Fig.3 and quantified in the following lemma.

Lemma 1. $u'_i \geq 4u_i$ for any \mathbf{x}_i , d_i , \mathbf{W} , and \mathbf{b} .

The proof of Lemma 1 is given in the Appendix.

Replacing u_i with $u'_i/4$ in the loss function Eq. (5) will generate an insensitive zone around the sigmoid function as represented by the shaded area in Fig. 3. Note that the vertical distance (along the d axis) from the two boundaries of the insensitive zone to the sigmoid curve might be different. This will result in an insensitive zone in the bivariate output space as represented by the shaded area in Fig. 2. It can be derived from Lemma 1 that, in Fig. 2, $0 < u_+^j \leq \varepsilon$ and $0 < u_-^j \leq \varepsilon$ for $j = 1, 2$. Thus, although not strictly isotropic, the shaded area is a reasonable approximation to the ideal hyper-spherical insensitive zone (the blue circle) so long as ε is small.

Replacing u_i with $u'_i/4$ in Eq. (2) yields a new target function $\Gamma'(\mathbf{W}, \mathbf{b})$. It is trivial to prove the following theorem with Lemma 1.

Theorem 1. $\Gamma'(\mathbf{W}, \mathbf{b})$ is an upper bound for $\Gamma(\mathbf{W}, \mathbf{b})$.

Therefore, minimizing $\Gamma'(\mathbf{W}, \mathbf{b})$ is equivalent to minimizing an upper bound of $\Gamma(\mathbf{W}, \mathbf{b})$.

It is still hard to minimize $\Gamma'(\mathbf{W}, \mathbf{b})$ as standard SVR does via solving its dual problem. Instead, we directly solve the primal problem with an iterative quasi-Newton method called Iterative Re-Weighted Least Square (IRWLS) [Pérez-Cruz *et al.*, 2000]. Firstly, $\Gamma'(\mathbf{W}, \mathbf{b})$ is approximated by its first order Taylor expansion at the solution of the current k -th iteration,

denoted by $\mathbf{W}^{(k)}$ and $\mathbf{b}^{(k)}$:

$$\Gamma''(\mathbf{W}, \mathbf{b}) = \frac{1}{2} \sum_{j=1}^c \|\mathbf{w}^j\|^2 + C \sum_{i=1}^n \left[L\left(\frac{u'_i{}^{(k)}}{4}\right) + \frac{dL(u')}{du'} \Big|_{\frac{u'_i{}^{(k)}}{4}} \frac{(e'_i{}^{(k)})^T}{4u'_i{}^{(k)}} (e'_i - e'_i{}^{(k)}) \right], \quad (10)$$

where $e'_i{}^{(k)}$ and $u'_i{}^{(k)}$ are calculated from $\mathbf{W}^{(k)}$ and $\mathbf{b}^{(k)}$. Then, a quadratic approximation is further constructed from Eq. (10):

$$\Gamma'''(\mathbf{W}, \mathbf{b}) = \frac{1}{2} \sum_{j=1}^c \|\mathbf{w}^j\|^2 + C \sum_{i=1}^n \left[L\left(\frac{u'_i{}^{(k)}}{4}\right) + \frac{dL(u')}{du'} \Big|_{\frac{u'_i{}^{(k)}}{4}} \frac{u'_i{}^{(k)} - (u'_i{}^{(k)})^2}{4u'_i{}^{(k)}} \right] \quad (11)$$

$$= \frac{1}{2} \sum_{j=1}^c \|\mathbf{w}^j\|^2 + \frac{1}{2} \sum_{i=1}^n a_i u_i^2 + \tau,$$

where

$$a_i = \frac{C}{2u'_i{}^{(k)}} \frac{dL(u')}{du'} \Big|_{\frac{u'_i{}^{(k)}}{4}} = \begin{cases} 0 & u'_i{}^{(k)} < 4\varepsilon, \\ \frac{C(u'_i{}^{(k)} - 4\varepsilon)}{4u'_i{}^{(k)}} & u'_i{}^{(k)} \geq 4\varepsilon, \end{cases} \quad (12)$$

and τ is a constant term that does not depend on \mathbf{W} and \mathbf{b} . Eq. (11) is a weighted least square problem whose optimum can be effectively found by letting the gradient equal zero and then solving a system of linear equations for $j = 1, \dots, c$:

$$\begin{bmatrix} \Phi^T D_a \Phi + \mathbf{I} & \Phi^T \mathbf{a} \\ \mathbf{a}^T \Phi & \mathbf{1}^T \mathbf{a} \end{bmatrix} \begin{bmatrix} \mathbf{w}^j \\ b^j \end{bmatrix} = \begin{bmatrix} -\Phi^T D_a \log\left(\frac{1}{d^j} - 1\right) \\ -\mathbf{a}^T \log\left(\frac{1}{d^j} - 1\right) \end{bmatrix}, \quad (13)$$

where $\Phi = [\varphi(\mathbf{x}_1), \dots, \varphi(\mathbf{x}_n)]^T$, $\mathbf{a} = [a_1, \dots, a_n]^T$, $(D_a)_{ij} = a_i \delta_{ij}$ (δ_{ij} is the Kronecker's delta function), $d^j = [d_1^j, \dots, d_n^j]^T$, $\mathbf{1}$ is a vector of all ones. Then, the direction of the optimal solution of Eq. (11) is used as the descending direction for the optimization of $\Gamma'(\mathbf{W}, \mathbf{b})$, and the solution for the next iteration ($\mathbf{W}^{(k+1)}$ and $\mathbf{b}^{(k+1)}$) is obtained via a line search algorithm [Nocedal and Wright, 2006] along this direction.

According to the Representer Theorem [Schlköpf and Smola, 2001], \mathbf{w}^j may be represented as a linear combination of the training examples in the feature space, i.e., $\mathbf{w}^j = \Phi^T \beta^j$. Substituting this expression into Eq. (13) yields

$$\begin{bmatrix} \mathbf{K} + D_a^{-1} & \mathbf{1} \\ \mathbf{a}^T \mathbf{K} & \mathbf{1}^T \mathbf{a} \end{bmatrix} \begin{bmatrix} \beta^j \\ b^j \end{bmatrix} = \begin{bmatrix} -\log\left(\frac{1}{d^j} - 1\right) \\ -\mathbf{a}^T \log\left(\frac{1}{d^j} - 1\right) \end{bmatrix}, \quad (14)$$

where $\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) = \varphi^T(\mathbf{x}_i)\varphi(\mathbf{x}_j)$ is the kernel matrix. Accordingly, the later line search for $\Gamma'(\mathbf{W}, \mathbf{b})$ can be performed in terms of β^j . Finally, after the optimal solution $\hat{\beta}^j$ is obtained, $\hat{\mathbf{w}}^j = \Phi^T \hat{\beta}^j$ and \hat{b}^j are substituted into Eq. (1) and the label distribution can be calculated indirectly in the original input space \mathcal{X} , rather than the high-dimensional feature space $\mathbb{R}^{\mathcal{H}}$, via the kernel matrix \mathbf{K} .

Table 1: Pre-release Metadata Included in the Data Set

Attribute	Type	θ	#Values
Genre	C	0	24
Color	C	0	2
Director	C	5	402
1st Actor	C	5	386
2nd Actor	C	5	210
3rd Actor	C	5	103
Country	C	5	33
Language	C	5	23
Writer	C	10	16
Editor	C	10	115
Cinematographer	C	10	173
Art Direction	C	10	39
Costume Designer	C	10	110
Music By	C	10	157
Sound	C	10	26
Production Company	C	20	31
Year	N	-	-
Running Time	N	-	-
Budget	N	-	-

4 Experiments

4.1 Methodology

The data set used in the experiments includes 7,755 movies and 54,242,292 ratings from 478,656 different users. The ratings come from Netflix, which are on a scale from 1 to 5 integral stars. Each movie has, on average, 6,994 ratings. The rating distribution is calculated for each movie as an indicator for the crowd opinion on that movie. The pre-release metadata are crawled from IMDb according to the unique movie IDs. Table 1 lists all the metadata included in the data set. Note that all the attributes in Table 1 can be retrieved before the movie is officially released or even during its planning phase. No post-release attributes are included in the data set, although some of them might be closely related to the crowd opinion, such as the box office gross. There are both numeric (N) and categorical (C) attributes in this data set. Some categorical attributes, typically human names, might have too many different values. In such case, we set a threshold θ and re-assign a new value ‘other’ to those who appear less times in the data set than the threshold. This will filter out most categories with limited influence on the crowd opinion, e.g., unfamous actors. The categorical attributes are then transformed into numeric ones by replacing the k -valued categorical attribute by k binary attributes, one for each value indicating whether the attribute has that value or not. Finally, all the attributes are normalized to the same scale through the min-max normalization.

As mentioned in Section 3, LDSVR deals with two challenges simultaneously: (1) *multivariate output* and (2) *probability output*. In order to show the advantages of solving these two problems simultaneously, LDSVR is compared with two baseline variants. The first is to fit a sigmoid function to each dimension separately, i.e., replace w_i^j in Eq. (3) by the absolute value of the j -th element of e_i^j calculated by Eq. (9), and then use $L_h(|e_i^j|)$ as the loss function. This variant is named as S-SVR standing for Sigmoid SVR, which solves the chal-

lenge (2) but not (1). The second variant is to firstly run a standard M-SVR [Fernández *et al.*, 2004], and then perform a post-process where the outputs are subtracted by a common bias determined by the minimum regression output over the training set and then divided by the sum of all elements. This variant is named as M-SVR_p standing for M-SVR plus post-process, which solves the challenge (1) but not (2). Note that the output of all regression methods should be finally normalized by dividing with the sum of all elements to make sure that $\sum_j d^j = 1$. This is reasonable in most cases where only the relative relationship matters in the distribution. LDSVR is also compared with existing typical LDL algorithms including BFGS-LLD [Geng and Ji, 2013], IIS-LLD [Geng *et al.*, 2010], AA- k NN [Geng and Ji, 2013], and CPNN [Geng *et al.*, 2013].

The performance of the algorithms is evaluated by those commonly used measures in LDL, i.e., the average distance or similarity between the predicted and ground truth label distributions. As suggested in [Geng and Ji, 2013], six measures are used in the experiments, which include four distance measures (the smaller the better), i.e., Euclidean, Sørensen, Squared χ^2 , and Kullback-Leibler (K-L), and two similarity measures (the larger the better), i.e., Intersection and Fidelity.

The algorithm parameters used in the experiments are empirically determined. The parameter selection process is nested into the 10-fold cross validation. In detail, the whole data set is first randomly split into 10 chunks. Each time, one chunk is used as the test set, another is used as the validation set, and the rest 8 chunks are used as the training set. Then, the model is trained with different parameter settings on the training set and tested on the validation set. This procedure is repeated 10 folds, and the parameter setting with the best average performance is selected. After that, the original validation set is merged into the training set and the test set remains unchanged. The model is trained with the selected parameter setting on the updated training set and tested on the test set. This procedure is repeated 10 folds and the mean value and standard deviation of each evaluation measure is reported. The final parameter settings for the compared algorithms are as follows. All kernel based methods (LDSVR, S-SVR and M-SVR_p) use the RBF kernel with the scaling factor σ equal to the average distance among the training examples. The penalty parameter C in Eq. (2) is set to 1. The insensitivity parameter ε is set to 0.1. All iterative algorithms terminate their iteration when the difference between adjacent steps is smaller than 10^{-10} . The number of neighbors k in AA- k NN is set to 10, and the number of hidden neurons in CPNN is set to 80.

4.2 Results

The experimental results of the seven algorithms on the movie data set are tabulated in Table 2. For the four distance measures (Euclidean, Sørensen, Squared χ^2 , and K-L), ‘↓’ indicates ‘the smaller the better’. For the two similarity measures (Intersection and Fidelity), ‘↑’ indicates ‘the larger the better’. The best performance on each measure is highlighted by boldface. On each measure, the algorithms are ranked in decreasing order of their performance. The ranks are given in the brackets right after the measure values.

Table 2: Experimental Results (mean±std(rank)) of the Seven Compared Algorithms

	Euclidean ↓	Sørensen ↓	Squared χ^2 ↓	K-L ↓	Intersection ↑	Fidelity ↑
LDSVR	.1587±.0026(1)	.1564±.0027(1)	.0887±.0031(1)	.0921±.0035(1)	.8436±.0027(1)	.9764±.0010(1)
S-SVR	.1734±.0024(2)	.1723±.0023(2)	.1040±.0030(2)	.1059±.0036(2)	.8277±.0023(2)	.9722±.0009(2)
M-SVR _p	.1843±.0031(3)	.1814±.0034(3.5)	.1084±.0033(3)	.1073±.0030(3)	.8186±.0034(3.5)	.9710±.0010(3)
BFGS-LLD	.1853±.0033(4)	.1814±.0033(3.5)	.1176±.0042(4)	.1265±.0050(4)	.8186±.0033(3.5)	.9683±.0012(4)
IIS-LLD	.1866±.0041(5)	.1828±.0044(5)	.1195±.0054(5)	.1288±.0070(6)	.8172±.0044(5)	.9676±.0014(5)
AA- <i>k</i> NN	.1917±.0045(6)	.1899±.0047(6)	.1246±.0062(6)	.1274±.0069(5)	.8101±.0047(6)	.9664±.0018(6)
CPNN	.2209±.0148(7)	.2153±.0150(7)	.1625±.0206(7)	.1826±.0274(7)	.7847±.0150(7)	.9551±.0061(7)

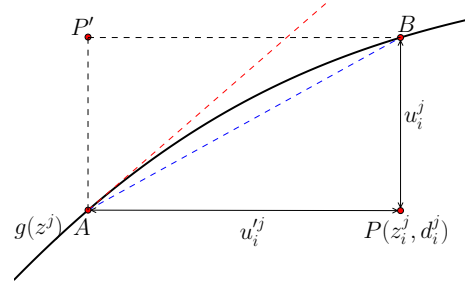
As can be seen from Table 2, LDSVR performs best on all of the six measures. The two variants of LDSVR, S-SVR and M-SVR_p, both perform significantly worse than LDSVR. This proves the advantage of LDSVR gained by means of solving the *multivariate output* problem and the *probabilistic output* problem simultaneously, rather than one by one. The performance of LDSVR is also significantly better than that of the compared LDL algorithms (BFGS-LLD, IIS-LLD, AA-*k*NN, and CPNN). The reason might be two-fold. First, while most existing LDL algorithms seek to directly minimize the K-L divergence between the predicted and ground truth distributions, LDSVR takes advantage of the large margin regression by a support vector machine. Second, application of the kernel trick makes it possible for LDSVR to solve the problem in a higher-dimensional thus more discriminative feature space without loss of computational feasibility.

5 Conclusion and Discussion

This paper investigates possibilities to predict the crowd opinion about a movie before it is released. The crowd opinion is represented by the distribution of ratings given by a sufficient number of people who have watched the movie. The pre-release prediction of the crowd opinion could be a crucial indicator of whether the movie will be successful or not, and thus has vast potential applications in movie production and marketing.

This paper regards the pre-release crowd opinion prediction as a Label Distribution Learning (LDL) problem, and proposes a Label Distribution Support Vector Regressor (LDSVR) for it. The nature of user rating distribution requires the output of LDSVR to be both multivariate and probabilistic. Thus, the basic idea of LDSVR is to fit a sigmoid function to each component of the rating distribution simultaneously by a multi-output support vector machine. Experiments on a data set including 7,755 movies reveal that LDSVR can perform significantly better than not only its two variants, but also four typical LDL algorithms.

One of the key ideas of LDSVR is to measure the loss by the distance needed to move in the input space to get the same output as the ground truth. This works fine for most cases, but sometimes could be risky when the ground truth outputs of some training examples are very close to 0 or 1. In such case, the loss of those examples will tend to be so large that they will dominate the training process. Although this is not a big problem for the movie rating distributions so long as there are a sufficient number of ratings for each movie, it might be


 Figure 4: The relationship between u_i^j and u_i^{tj} in the concave part of the sigmoid.

troublesome for some other data. Therefore, a mechanism of compensating the loss of those examples with outputs close to 0 or 1 might be an important future work to make LDSVR applicable to more general cases.

Appendix: Proof of Lemma 1

Proof. Firstly, consider the situation in the (z^j, d^j) -space ($j = 1, \dots, c$), where z^j and d^j represent the j -th dimension of \mathbf{z} and \mathbf{d} , respectively. The projections of u_i and u_i' in such space are denoted by u_i^j and $u_i'^j$, respectively. Since $u_i = \sqrt{\sum_j (u_i^j)^2}$ and $u_i' = \sqrt{\sum_j (u_i'^j)^2}$, if we can prove $u_i'^j \geq 4u_i^j$ for $j = 1, \dots, c$, then we have $u_i' \geq 4u_i$.

In the (z^j, d^j) -space, the sigmoid function $d^j = g(z^j)$ is symmetric about the point $(0, 0.5)$. So we only need to consider the case $z^j \geq 0$, where $g(z^j)$ is concave as shown in Fig. 4. When the point is above the curve like P' , we can always find its counterpart P below the curve with same vertical and horizontal distance to the curve. Thus, we only need to consider the case when the point is below the curve.

Since $g(z^j)$ is concave, the line segment AB (blue dash line) is always below the curve. Suppose its slope is θ , and the slope of the $g(z^j)$ curve's tangent line (red dash line) at the point A is θ' , then $\theta \leq \theta'$. The derivative of $g(z^j)$ is

$$g'(z^j) = \frac{\exp(-z^j)}{(1 + \exp(-z^j))^2}. \quad (15)$$

Letting the second derivative of $g(z^j)$ equal zero yields $z^j = 0$. Thus, the maximum of θ' is $g'(0) = 1/4$. Therefore,

$$\theta = u_i^j / u_i'^j \leq \theta' \leq 1/4. \quad (16)$$

So $u_i'^j \geq 4u_i^j$, and thus $u_i' \geq 4u_i$. \square

References

- [Adomavicius and Tuzhilin, 2005] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowl. Data Eng.*, 17(6):734–749, 2005.
- [Amolochitis *et al.*, 2014] Emmanouil Amolochitis, Ioannis T. Christou, and Zheng-Hua Tan. Implementing a commercial-strength parallel hybrid movie recommendation engine. *IEEE Intelligent Systems*, 29(2):92–96, 2014.
- [Bresler *et al.*, 2014] Guy Bresler, George H. Chen, and Devavrat Shah. A latent source model for online collaborative filtering. In *Advances in Neural Information Processing Systems 27 (NIPS'14)*, pages 3347–3355, Montreal, Canada, 2014.
- [Diao *et al.*, 2014] Qiming Diao, Minghui Qiu, Chao-Yuan Wu, Alexander J. Smola, Jing Jiang, and Chong Wang. Jointly modeling aspects, ratings and sentiments for movie recommendation (JMARS). In *Proc. 20th Int'l Conf. Knowledge Discovery and Data Mining*, pages 193–202, New York, NY, 2014.
- [Drucker *et al.*, 1996] Harris Drucker, Christopher J. C. Burges, Linda Kaufman, Alex J. Smola, and Vladimir Vapnik. Support vector regression machines. In *Advances in Neural Information Processing Systems 9 (NIPS'96)*, pages 155–161, Denver, CO, 1996.
- [Fernández *et al.*, 2004] Matilde Sánchez Fernández, Mario de Prado-Cumplido, Jerónimo Arenas-García, and Fernando Pérez-Cruz. SVM multiregression for nonlinear channel estimation in multiple-input multiple-output systems. *IEEE Trans. Signal Processing*, 52(8):2298–2307, 2004.
- [Geng and Ji, 2013] Xin Geng and Rongzi Ji. Label distribution learning. In *Proc. 13th IEEE Int'l Conf. Data Mining Workshops*, pages 377–383, Dallas, TX, 2013.
- [Geng and Luo, 2014] Xin Geng and Longrun Luo. Multi-label ranking with inconsistent rankers. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 3742–3747, Columbus, OH, 2014.
- [Geng and Xia, 2014] Xin Geng and Yu Xia. Head pose estimation based on multivariate label distribution. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 3742–3747, Columbus, OH, 2014.
- [Geng *et al.*, 2010] Xin Geng, Kate Smith-Miles, and Zhi-Hua Zhou. Facial age estimation by learning from label distributions. In *Proc. 24th AAAI Conf. Artificial Intelligence*, pages 451–456, Atlanta, GA, 2010.
- [Geng *et al.*, 2013] Xin Geng, Chao Yin, and Zhi-Hua Zhou. Facial age estimation by learning from label distributions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(10):2401–2412, 2013.
- [Jin *et al.*, 2005] Xin Jin, Yanzan Zhou, and Bamshad Mobasher. A maximum entropy web recommendation system: combining collaborative and content features. In *Proc. 11th Int'l Conf. Knowledge Discovery and Data Mining*, pages 612–617, Chicago, IL, 2005.
- [Marovic *et al.*, 2011] Mladen Marovic, Marko Mihokovic, Mladen Miksa, Sinisa Pribil, and Alan Tus. Automatic movie ratings prediction using machine learning. In *Proc. 34th Int'l Conv. Information and Communication Technology, Electronics and Microelectronics*, pages 1640–1645, Opatija, Croatia, 2011.
- [Nocedal and Wright, 2006] Jorge Nocedal and Stephen Wright. *Numerical Optimization*. Springer, New York, NY, 2nd edition, 2006.
- [Pérez-Cruz *et al.*, 2000] Fernando Pérez-Cruz, Pedro Luis Alarcón-Diana, Angel Navia-Vázquez, and Antonio Artés-Rodríguez. Fast training of support vector classifiers. In *Advances in Neural Information Processing Systems 13 (NIPS'00)*, pages 734–740, Denver, CO, 2000.
- [Pérez-Cruz *et al.*, 2002] Fernando Pérez-Cruz, Gustavo Camps-Valls, Emilio Soria-Olivas, Juan José Pérez-Ruixo, Aníbal R. Figueiras-Vidal, and Antonio Artés-Rodríguez. Multi-dimensional function approximation and regression estimation. In *Proc. Int'l Conf. Artificial Neural Networks*, pages 757–762, Madrid, Spain, 2002.
- [Pietra *et al.*, 1997] Stephen Della Pietra, Vincent J. Della Pietra, and John D. Lafferty. Inducing features of random fields. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(4):380–393, 1997.
- [Pilászy and Tikk, 2009] István Pilászy and Domonkos Tikk. Recommending new movies: even a few ratings are more valuable than metadata. In *Proc. 2009 ACM Conference on Recommender Systems*, pages 93–100, New York, NY, 2009.
- [Platt, 1999] John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In Alexander J. Smola, Peter J. Bartlett, Bernhard Schölkopf, and Dale Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, 1999.
- [Schlköpf and Smola, 2001] Bernhard Schlköpf and Alexander J. Smola. *Learning with Kernels*. The MIT Press, Cambridge, MA, 2nd edition, 2001.
- [Soares and Viana, 2014] Márcio Soares and Paula Viana. Tuning metadata for better movie content-based recommendation systems. *Multimedia Tools and Applications*, Published online, 2014.
- [Tsochantaridis *et al.*, 2004] Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support vector machine learning for interdependent and structured output spaces. In *Proc. 21st Int'l Conf. Machine Learning*, Banff, Canada, 2004.