Generalized Optimal Reverse Prediction

Martha White and Dale Schuurmans
Department of Computing Science
University of Alberta
{whitem, dale}@cs.ualberta.ca

Abstract

Recently it has been shown that classical supervised and unsupervised training methods can be unified as special cases of so-called "optimal reverse prediction": predicting inputs from target labels while optimizing over both model parameters and missing labels. Although this perspective establishes links between classical training principles, the existing formulation only applies to linear predictors under squared loss, hence is extremely limited. We generalize the formulation of optimal reverse prediction to arbitrary Bregman divergences, and more importantly to non-linear predictors. This extension is achieved by establishing a generalized form of forward-reverse minimization equivalence that holds for arbitrary matching losses. Several benefits follow. First, a new variant of Bregman divergence clustering can be recovered that incorporates a non-linear data reconstruction model. Second, normalized-cut and kernel-based extensions can be formulated coherently. Finally, a new semi-supervised training principle can be recovered for classification problems that demonstrates some advantages over the state of the art.

1 Introduction

Unsupervised learning has a long history in machine learning and statistics, focusing on problems of discovering latent structure in data, such as clusters or manifolds. Supervised learning has an equally long history, but is normally considered a distinct learning problem, with separate data requirements and training principles. For example, dimensionality reduction and clustering methods classically rely on principles for re-representing input data, whereas classical regression and classification methods seek to minimize prediction error on associated output variables. Although minimizing prediction error on outputs is not completely irrelevant to input data reconstruction, the two are distinct processes and a growing set of separate principles have been developed for each. Even in a purely probabilistic setting a similar diversity of training principles exist, particularly for "discriminative" models [Lasserre *et al.*, 2006; Druck *et al.*, 2007; Druck and McCallum, 2010b].

Unfortunately, the separation between unsupervised and supervised training principles cannot be maintained in the context of *semi-supervised* learning. Here one must train on a mix of labeled and unlabeled data and therefore reconcile separate supervised and unsupervised training criteria in a common framework. Given the diversity of supervised and unsupervised training principles, it is often unclear how they can best be combined. In fact, there remains a proliferation of proposed approaches [Zhu, 2006] with essentially no theoretical guarantee that exploiting unlabeled data will even avoid harm [Ben-David *et al.*, 2008; Li and Zhou, 2011; Nadler *et al.*, 2009]. Recently, some diminishment of research activity on semi-supervised learning has taken place, with the dominant approaches still being (i) to use a supervised loss with an unsupervised loss regularizer, which includes many graph-based methods [Belkin *et al.*, 2006; Corduneanu and Jaakkola, 2006; Zhou *et al.*, 2004]; (ii) to combine self-supervised training on the unlabeled data with supervised on the labeled [Bie and Cristianini, 2003; Joachims, 1999]; (iii) to train a joint probability model generatively [Bishop, 2006; Druck and McCallum, 2010a; Nigam *et al.*, 2000]; and (iv) co-training [Blum and Mitchell, 1998]. In all cases, supporting theory has been slower to develop, with few exceptions [Balcan and Blum, 2005; Ben-David *et al.*, 2008].

Xu et al. Xu *et al.* [2009] provide another perspective on the problem by first reformulating supervised and unsupervised training principles in a common framework before combining them in a semi-supervised learning method. [Xu *et al.*, 2009] demonstrated that several classical supervised and unsupervised training methods, such as least

Appearing in Proceedings of the 15th International Conference on Artificial Intelligence and Statistics (AISTATS) 2012, La Palma, Canary Islands. Volume XX of JMLR: W&CP XX. Copyright 2012 by the authors.

squares regression, principal components analysis, and k-means clustering, could be equivalently expressed as so-called "optimal reverse prediction"—predicting inputs from targets while optimizing over model parameters and missing labels. Unfortunately, these unifications were achieved in a very limited setting by restricting attention to linear predictors and squared loss. The framework did however yield a semi-supervised learning approach with a non-trivial guarantee that additional unlabeled data would help rather than harm the training process.

In this paper we demonstrate that the unification proposed by [Xu *et al.*, 2009] need not be restricted in such a way. In particular, we re-express optimal reverse prediction in a general manner that admits arbitrary Bregman divergence losses and non-linear transfer functions via matching losses [Helmbold *et al.*, 2002]. We show that this extension allows: (1) some new unsupervised training principles to be derived, such as Bregman divergence based clustering with non-linear data reconstruction; (2) standard extensions such as kernels and normalized graph-cut criteria to be coherently achieved; and (3) a simple new semi-supervised learning algorithm to be devised that combines supervised and unsupervised losses in a principled manner, achieving a very similar guarantee about the benefits of additional unlabeled data for learning.

Below, we first formulate reverse prediction with non-linear transfer functions and use *matching losses* to recover convex loss functions for a given non-linear transfer. We then demonstrate how to to incorporate regularization, kernels and instance weighting into the generalized framework. We prove that two well known methods, exponential family principal component analysis [Collins *et al.*, 2002; Gordon, 2003; Canu and Smola, 2006] and clustering with Bregman divergences [Banerjee *et al.*, 2005], can be expressed as special cases of the generalized framework. Along the way we identify several algorithmic corrections and extensions: (1) that the original exponential family principal component analysis formulation was not conditionally convex (now providing an equivalent, but conditionally convex formulation using reverse prediction); (2) how non-linear transfers can be applied to clustering with Bregman divergences (which currently only allows linear models); (3) kernelized exponential family principal component analysis; and (4) normalized cut using Bregman divergences. We present a simple semi-supervised algorithm that generalizes Xu et al's. [2009] algorithm. Interestingly, we are able to prove a similar variance reduction result that unlabeled data reduces variance in the error estimate of the reverse model. Finally, we empirically demonstrate the advantages of having non-linear transfer functions and a principled objective for semi-supervised learning.

2 Preliminaries

To begin, we consider a simple supervised learning set-up. Assume one is given a $t \times n$ input data matrix, X, with rows corresponding to instances and columns to features, and corresponding $t \times k$ output (label) matrix, Y. Furthermore, assume data are observed as independent rows, hence we use X_i to denote the ith observation (row) in X, and Y_i to denote the ith output (row) in Y. Finally, we adopt a suitable transfer function $\mathbf{f}$ that emits non-linear predictions $\hat{Y} = \mathbf{f}(XW)$, where $\mathbf{f}$ is applied row-wise.

To formulate a supervised training problem, we need to select a training loss, L. Since the data are observed as independent rows, the training loss will be expressed as a row-wise summation, written in shorthand as $L(XW, Y) = \sum_i L(X_i W, Y_i)$. Clearly, the choice of loss function can be critical. For example, for a sigmoid transfer function, $f(X_i W) = 1/(1 + e^{-X_i W})$ (mapping a row vector to a row vector), the least squares loss, $\sum_i \|f(X_i W) - Y_i\|_2^2$ can have exponentially many local minima [Auer *et al.*, 1996]. In general, choosing one loss function for any transfer function will likely result in a nonconvex minimization.

In the context of prediction with non-linear transfer functions, a well known solution to this problem is to use a *matching loss* [Gentile and Warmuth, 1999; Helmbold *et al.*, 2002; Kivinen and Warmuth, 2001]. That is, for any *strictly* convex potential function $F : \mathbb{R}^r \to \mathbb{R}$, where $\mathbf{f} = \nabla F$ is the associated transfer function, the corresponding matching loss between a pre-prediction vector $\hat{\mathbf{z}} = \mathbf{x}\hat{W}$ and a target vector $\mathbf{y} = f(\mathbf{z})$ for some $\mathbf{z}$ is given by

$$L(\hat{\mathbf{z}}, \mathbf{y}) = F(\hat{\mathbf{z}}) - F(\mathbf{z}) - f(\mathbf{z})^T(\hat{\mathbf{z}} - \mathbf{z}) = D_F(\hat{\mathbf{z}}||\mathbf{z})$$

where D_F is the Bregman divergence for potential function F. Since the loss is a Bregman divergence, it has many useful properties, including (i) $L(\hat{\mathbf{z}}, \mathbf{y}) = 0 \Leftrightarrow \hat{\mathbf{z}} = f^{-1}(\mathbf{y})$; (ii) $L(\hat{\mathbf{z}}, \mathbf{y}) \geq 0$; and (iii) L is convex in the first argument. Matching losses encompass a wide range of losses, including least squares (identity transfer), cross entropy (sigmoidal transfer) and relative entropy (softmax transfer) [Kivinen and Warmuth, 2001]. Given row-wise independence, we can denote the matching loss as

$$L(XW, Y) = D_F(XW||f^{-1}(Y)) = \tag{1}$$
$$\sum_{i=1}^{t} F(\mathbf{f}^{-1}(Y_i)) - F(X_i W) - Y_i(X_i W - \mathbf{f}^{-1}(Y_i))^T$$

and the corresponding loss minimization problem as

$$\min_W \; L(XW, Y) \equiv \min_W \sum_{i=1}^{t} F(X_i W) - Y_i W^T X_i^T. \tag{2}$$

Thus, in standard supervised learning, non-linear transfers significantly extend modelling power, while matching losses allow convex training criteria to be recovered for any transfer that is the gradient of a convex potential.

3 Reverse Prediction with Matching Losses

The first result is that an equivalence can be established between distinct "forward" and "reverse" supervised training problems, even in the presence of non-linear transfers defined by potentials. The forward problem seeks to predict the outputs Y from X, while conversely, the reverse problem considers predicting the inputs X from Y.

To relate these two problems we need the following definitions. For a strictly convex potential F, let $F^*(\mathbf{y}) = \sup_{\mathbf{z}} \mathbf{z}^T\mathbf{y} - F(\mathbf{z})$ denote its Fenchel conjugate. Intuitively, the value of the Fenchel conjugate at a point $\mathbf{y}$ is the intercept of the line with slope $\mathbf{y}$ that is tanget to F, which must be unique in this case by the strict convexity of F. The associated transfer function, $\mathbf{f}^* = \nabla F^*$, satisfies $\mathbf{f}^* = \mathbf{f}^{-1}$ (Lemma 4 in the Appendix). Since the forward loss can be written $L(XW, Y) = \sum_i D_F(X_iW||\mathbf{f}^*(Y_i))$, the corresponding reverse loss for the transfer, $\mathbf{f}$, can be written

$$R(X, YU) = D_{F^*}(YU||\mathbf{f}(X)) = \tag{3}$$

$$\sum_{i=1}^{t} F^*(Y_iU) - F^*(\mathbf{f}(X_i)) - X_i(Y_iU - \mathbf{f}(X_i))^T \tag{4}$$

yielding the corresponding reverse loss minimization

$$\min_U R(X, YU) \equiv \min_U \sum_{i=1}^{t} F^*(Y_iU) - X_iU^TY_i^T. \tag{5}$$

Although the reverse training problem (5) is not identical to the forward training problem (2), one can establish a unique correspondence between their optimal solutions; that is, given U^* one can uniquely recover W^* and vice versa. For the special case of an identity transfer $\mathbf{f}(\mathbf{z}) = \mathbf{z}$ and corresponding squared loss, Xu *et al.* [2009] have already demonstrated that a unique correspondence exists between the minimizers of (2) and (5). However, that result was established by a reduction to linear algebraic properties that are no longer available. Instead we generalize the result by showing that the correspondence can still be established between the unique optimal solutions to the forward and reverse problems under an arbitrary matching loss with a non-linear transfer $\mathbf{f}$.

Theorem 1 (Forward-Reverse Equivalence) *Given a $t \times n$ input matrix X and a $t \times k$ output matrix Y, such that $t >$ rank$(X) = n$ and $t >$ rank$(Y) = k$, there exist unique global minimizers*

$$W^* = \arg\min_W D_F(XW||\mathbf{f}^{-1}(Y)) \tag{6}$$

$$U^* = \arg\min_U D_{F^*}(YU||\mathbf{f}(X)) \tag{7}$$

$$\textit{where} \quad X^T\mathbf{f}(XW^*) = X^TY = \mathbf{f}^{-1}(YU^*)^TY. \tag{8}$$

(Proof given in the Appendix.[1]) Although in general the relationship between W^* and U^* is implicit, in some cases an explicit conversion can be recovered.

[1]All proofs not in the main body are given in the Appendix.

Corollary 2 *If X has full row rank, then*

$$W^* = X^{-1}\mathbf{f}^{-1}(X^{-T}\mathbf{f}^{-1}(YU^*)^TY).$$

This reverse loss minimization problem will play a key role in formulating *unsupervised* training principles, and relating them to the stated supervised training principles. Before addressing unsupervised learning in detail in Section 4, we first demonstrate that the generalized form of reverse prediction still admits standard regularizers, kernels, and instance weighting.

3.1 Extensions

Regularization, kernels and instance weighting are widely-used extensions that are essential for effective practical performance of learning algorithms. In particular, they aide overfitting avoidance, extend modeling power, and enable appropriate emphasis to be placed on data points, respectively. It is not obvious, however, whether all three of these extensions can be accommodated within the reverse training framework while preserving solution equivalence to forward training, as established above. Fortunately, by exploiting the relationship (8), one can show that all of the desired correspondences can in fact be preserved under such extensions.

First, note that for *regularization*, if one adds a strictly convex, differentiable regularizer, $R : \mathbb{R}^{n\times k}$, to the forward training problem

$$\arg\min_W D_F(XW||f^{-1}(Y)) + \alpha R(W) \tag{9}$$

one immediately obtains the solution equivalence

$$X^T\mathbf{f}(XW^*) + \alpha\nabla R(W^*) = \mathbf{f}^{-1}(YU^*)^TY \tag{10}$$

between (5) and (9). This allows an optimal W^* to be uniquely recovered from an optimal U^* under the same conditions as Theorem 1.

Second, re-expressing the training problem in a *reproducing kernel Hilbert space* (RKHS) is an important extension: kernels increase modeling power and enable a large, even infinite number of features. Note that one cannot simply incorporate a kernel into a non-linear transfer function: inner products, for example, are not invertible. However, from the representer theorem [Kimeldorf and Wahba, 1971], the regularized loss (9) must admit an RKHS embedding provided the regularizer is of the form $R = \Omega(h(W^TW))$ for a function h that is matrix nondecreasing [Argyriou *et al.*, 2009].[2] This form incorporates many useful regularizers, including the commonly used Frobenius norm ($\|W\|_F^2$)

[2]A function h is matrix non-decreasing if $M \succeq N$ implies $h(M) \geq h(N)$ for $M \succeq 0$ and $N \succeq 0$.

and trace norm($\|W\|_{tr}$). Thus, $\mathbf{x}^T W$ can be represented as $\sum_i \alpha_i k(\mathbf{x}_i, \mathbf{x})$ for some given values $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$. For a least squares regularizer, for example, since the form of $W = XA$, the kernelized form of Equation (9) becomes

$$\arg\min_A D_F(KA\|f^{-1}(Y)) + \alpha\,\mathrm{tr}(AA^T K). \tag{11}$$

Using the corresponding reverse loss $D_{F^*}(YB\|f(K))$ we obtain the equivalence

$$K^T \mathbf{f}(KA^*) + 2\alpha KA^* = \mathbf{f}^{-1}(YB^*)^T Y. \tag{12}$$

Finally, one can easily incorporate *instance weights* in the formulations while still preserving unique forward-reverse solution correspondences. Note that the Fenchel dual of $F_\lambda(x) = \lambda F(x)$ is given by $F^*_\lambda(y) = \lambda F^*(y/\lambda)$ for $\lambda > 0$, with corresponding gradients $\mathbf{f}_\lambda(\mathbf{x}) = \lambda \mathbf{f}(\mathbf{x})$ and $\mathbf{f}^*_\lambda(\mathbf{y}) = \mathbf{f}^{-1}(\mathbf{y}/\lambda)$. Despite the modified conjugates, the reverse problem simplifies to adding instance weights to the original reverse loss. To illustrate, consider an instance weighted version of the forward loss minimization problem

$$\begin{aligned}&\min_W \textstyle\sum_{i=1}^t \lambda_i D_F(X_i W\|\mathbf{f}^{-1}(Y_i))\\ &\quad = \min_W \textstyle\sum_{i=1}^t D_{F_{\lambda_i}}(X_i W\|\mathbf{f}_\lambda^{-1}(\lambda_i Y_i)).\end{aligned} \tag{13}$$

Using the above identity, we conclude that the corresponding reverse loss can be expressed

$$\begin{aligned}&D_{F^*_{\lambda_i}}(\lambda_i Y_i U_i\|\mathbf{f}_{\lambda_i}(X_i))\\ &\quad = \lambda_i F^*(\lambda_i Y_i U \lambda_i^{-1}) - \lambda_i F^*(\lambda_i \mathbf{f}(X_i)\lambda_i^{-1})\\ &\qquad - \mathbf{f}^{-1}(\lambda_i \mathbf{f}(X_i)\lambda_i^{-1})(\lambda_i Y_i U - \lambda_i \mathbf{f}(X_i))\\ &\quad = \lambda_i F^*(Y_i U) - \lambda_i F^*(\mathbf{f}(X_i))\\ &\qquad - \lambda_i \mathbf{f}^{-1}(\mathbf{f}(X_i))(Y_i U - \mathbf{f}(X_i)).\end{aligned}$$

This yields the same results as minimizing $\lambda_i D_{F^*}$; i.e.

$$\begin{aligned}&\min_U \textstyle\sum_{i=1}^t D_{F^*_{\lambda_i}}(\lambda_i Y_i U_i\|\mathbf{f}_\lambda(X_i))\\ &\quad \equiv \min_U \textstyle\sum_{i=1}^t \lambda_i D_{F^*}(Y_i U\|\mathbf{f}(X_i)\end{aligned} \tag{14}$$

therefore forward-reverse solution equivalence is retained.

4 Unsupervised Learning

In supervised learning, re-expressing standard forward training criteria in corresponding reverse formulations appears superfluous. However, for *unsupervised* learning the reverse formulation is necessary, since the forward optimization for unsupervised learning is vacuous, while the reverse optimization is not. Notice that for any W, Z can be set to $Z = \mathbf{f}(XW)$ to obtain $D_F(XW\|f^*(Z)) = 0$ (a vacuous result). The same is not true of U and Z in $D_{F^*}(ZU\|f(X))$, assuming $k < n$.

Importantly, the same forward-reverse equivalence can be maintained for unsupervised learning. By Theorem 1, first jointly learning a reverse model and labels, U^* and Z^*, fixing $Y = Z^*$ then optimizing the forward loss, is guaranteed to produce the unique optimal forward model, W^*, corresponding to Z^* and U^*. In this section, we exploit this generality to extend and improve existing unsupervised learning within the reverse prediction framework.

4.1 Exponential Family PCA

In standard prinicipal component analysis (PCA), one seeks to minimize the negative log likelihood between the data, X, and estimated parameters, $\boldsymbol{\theta}$, with the assumption that $P(\mathbf{x}|\boldsymbol{\theta})$ is normally distributed. In exponential family PCA, this distribution can instead be any regular exponential family (also called natural exponential family):

$$P_{F^*}(\mathbf{x}|\boldsymbol{\theta}) = \exp(\mathbf{x}^T\boldsymbol{\theta} - F^*(\boldsymbol{\theta}))p_0(\mathbf{x})$$

where F^* is commonly thought of as the cumulant function. Examples of natural exponential families include the Gaussian, gamma, chi-square, beta, Weibull, Bernoulli and Poisson distributions Banerjee *et al.* [2005]. Furthermore, many distributions can also be approximated with exponential families, greatly extending the applicability of PCA.

Collins et al. [2002] claimed (without proof) that this loss could be expressed through a Bregman distance:

$$\begin{aligned}&-\log P(\mathbf{x}|\boldsymbol{\theta}) = -\log P_0(\mathbf{x}) - F(\mathbf{x}) + D_F(\mathbf{x}\|\mathbf{f}^{-1}(\boldsymbol{\theta}))\\ &\Rightarrow \min_{\mathbf{z}}\min_U -\log(P(\mathbf{x}^T|\mathbf{z}^T U))\\ &\qquad\qquad \equiv \min_{\mathbf{z}}\min_U D_F(\mathbf{x}^T\|\mathbf{f}^{-1}(\mathbf{z}^T U))\end{aligned} \tag{15}$$

with $\boldsymbol{\theta}^T = \mathbf{z}^T U$. Here $\mathbf{z}$ is the low dimensional representation of $\mathbf{x}$ and U is the projection matrix onto the $\mathbf{z}$ space. The first equality results from the fact that $\mathbf{f}^{-1}(\boldsymbol{\theta})\boldsymbol{\theta} = F(\mathbf{f}^{-1}(\boldsymbol{\theta})) + F^*(\boldsymbol{\theta})$. Banerjee et al. [2005] proved that this relationship was valid for regular exponential families.

Notice that the optimization on $D_F(\mathbf{x}\|\mathbf{f}^{-1}(\boldsymbol{\theta}))$ is not necessarily convex, as Bregman divergences are only guaranteed to be convex in the first argument. In the following, we illustrate that exponential family PCA is an instance of reverse prediction, framed now as a convex optimization.

Definition 3 *A **regular exponential family distribution**, $P_{F^*}(\mathbf{x}|\theta)$, has $F^*(\mathbf{x}) = \ln(g(\mathbf{x}))$ with $g : X^n \to \mathbb{R}^+$ a continuous exponentially convex function with X open and F^* strictly convex. D_{F^*} is a **regular Bregman divergence**. A continuous function $g : X^n \to \mathbb{R}^+$ is exponentially convex function iff it is a Laplace transform of a nonnegative finite measure. Examples include $\exp(ax + bx^2)$, $(\exp(x) - 1)/x$, x^{-n} and $\sinh(x)/x$ [Ehm* et al.*, 2003].*

Lemma 4 *With $\hat{\mathbf{y}} = \mathbf{f}(\hat{\mathbf{z}})$ and $\mathbf{y} = \mathbf{f}(\mathbf{z})$,*

$$D_F(\hat{\mathbf{z}}\|\mathbf{z}) = D_{F^*}(\mathbf{y}\|\hat{\mathbf{y}}). \tag{16}$$

Theorem 5 *For D_{F^*} a regular Bregman divergence, then unconstrained reverse prediction*

$$\min_Z \min_U \; D_{F^*}(ZU||\mathbf{f}(X)) \tag{17}$$

is equivalent to regular exponential family principal components analysis (EPCA).

Proof: From Banerjee et al. [2005], we know that minimizing the log likelihood for regular exponential families corresponds to minimizing a regular Bregman divergence in Equation 15. This is not a convex minimization, but using Lemma 4, we convert the minimization to a dual space.

$$\begin{aligned} D_{F^*}(ZU||\mathbf{f}(X)) &= D_F(\mathbf{f}^{-1}(\mathbf{f}(X))||\mathbf{f}^{-1}(ZU)) \\ &= D_F(X||\mathbf{f}^{-1}(ZU)) \end{aligned}$$

because $(F^*)^* = F$. Therefore, the reverse prediction optimization is equivalent to EPCA. ■

Moreover, based on the previous kernel extensions for generalized reverse prediction, one can extend exponential family PCA to *kernel* exponential family PCA.

Corollary 6 *Kernelized reverse prediction with regular Bregman divergences*

$$\min_Z \min_U \; D_{F^*}(ZU||\mathbf{f}(K)) \tag{18}$$

is equivalent to kernel EPCA.

Thus far, we have illustrated how a generalized reverse prediction framework enables improvements and extensions for exponential family PCA. We next present some useful modifications to Bregman clustering using other insights offered by the framework.

4.2 Clustering with Bregman Divergences

Banerjee et al. [2005] generalized the centroid-based hard clustering problem to clustering using any Bregman divergence. We will once again find that Bregman clustering is an instance of reverse prediction, despite the fact that they reach their generalization using the idea of Bregman information rather than reverse prediction. More importantly, however, the formulation of Bregman clustering expressed in [Banerjee *et al.*, 2005] only permits *linear* transfers (as seen in Theorem 7). Under reverse prediction, we will be able to provide a new Bregman clustering formulation that incorporates non-linear transfers.

Theorem 7 *Constrained reverse prediction with any Bregman divergence and non-transfered ZU:*

$$\min_{Z: Z\in\{0,1\}^{t\times k},\, Z1=1} \; \min_U \; D_{F^*}(\mathbf{f}(ZU)||\mathbf{f}(X)) \tag{19}$$

is equivalent to clustering with Bregman divergences.

Proof: For $\mathcal{Z} = \{Z \in \{0,1\}^{t\times k} \mid Z1 = 1\}$, k-means clustering with Bregman divergences was framed under the minimization $\min_{Z\in\mathcal{Z}} \min_U D_F(X||ZU)$ [Banerjee *et al.*, 2005]. Thus, we can apply the same argument as in Th. 5. ■

Since Z is a discrete $\{0,1\}$ variable, Banerjee et al. [2005] show how the optimization for Bregman clustering can be simplified to a simple k-means algorithm. Using a similar insight, we can formulate a correspondingly efficient form of Bregman divergence clustering *that incorporates a non-linear transfer between ZU and Y*:

$$\begin{aligned} D_{F^*}(ZU||\mathbf{f}(X)) &= D_F(X||\mathbf{f}^*(ZU)) \\ &= D_F(X||Z\mathbf{f}^*(U)) \qquad (20) \\ \Longrightarrow \min_{Z\in\mathcal{Z}} & \min_{U\in Dom(\mathbf{f}^*)} D_{F^*}(ZU||\mathbf{f}(X)) \qquad (21) \\ = \min_{Z\in\mathcal{Z}} & \min_{M\in Dom(\mathbf{f})\subset\mathbb{R}^{n\times k}} D_{F^*}(X||ZM) \qquad (22) \\ = \min_{Z\in\mathcal{Z}} & \min_{M\in Dom(\mathbf{f})} \sum_{j=1}^{k} \sum_{i:Z_{ij}=1} D_{F^*}(X_{i:}||ZM_{:j}) \qquad (23) \\ = & \min_{Z\in\mathcal{Z}} \sum_j \frac{1}{\mathbf{1}^T Z_{:j}} \sum_{i:Z_{ij}} X_{i:}. \qquad (24) \end{aligned}$$

(The proof for the simplification of the inner maximization is given in the Appendix.) Algorithm 1 illustrates our modified Bregman clustering algorithm which now permits non-linear transfers.

Banerjee et al. [2005] also extend their algorithm to mixture models (relaxed constraints on $Z \in [0,1]$), by using regular Bregman divergences. Algorithm 2 gives the new objective and corresponding updates for our mixture model clustering algorithm with non-linear transfers, using similar arguments to those above. Note that as $\rho \to \infty$, the update converges to the hard clustering update.

As an important note, a popular algorithm for clustering speech data in the signal processing community (Linde-Buzo-Gray algorithm) and information-theoretic clustering are special cases of Bregman clustering depending on the choice of Bregman divergence [Banerjee *et al.*, 2005]. Therefore, the above unification and non-linear transfer results apply to these algorithms as well. It is important to notice that these reverse losses can either be used with or without a non-linear transfer: the Bregman divergence allows different distribution assumptions on the underlying data, but the modeling power can be reduced without the ability to use the non-linear transfers.

Theorem 8 *Probabilistic constrained reverse prediction with regular Bregman divergences and non-transferred ZU*

$$\min_{Z: Z\in[0,1]^{t\times k},\, Z1=1} \; \min_U \; D_{F^*}(\mathbf{f}(ZU)||\mathbf{f}(X)) \tag{25}$$

is equivalent to mixture model clustering with Breg. div.

Algorithm 1 ReverseBregmanHardClustering(X, k, D_F)

```
1: Initialize $M$ (e.g. $k$ randomly selected rows from $X$)
2: while (change in $D_F(X||M)$) $>$ tolerance do
3:     E-Step: $\forall i \in \{1, \dots, t\}$ :
4:         $Z(i, \arg\min_j D_F(X(i,:)||M(:,j))) = 1$
5:     M-Step: $\forall j \in \{1, \dots, k\}$ :
6:         $M_{:j} = \frac{1}{\mathbf{1}^T Z_{:j}} \sum_{i:Z_i j=1} X_{i:}$
7: end while
```

Algorithm 2 ReverseBregmanSoftClustering(X, k, D_f)

```
1: Initialize $M$ (e.g. $k$ randomly selected rows from $X$)
2: err$(M, p) = -\sum_i \log\left(\sum_j p_j \exp(-\rho D_F(X_{i:}||M_{:j}))\right)$
3: while (change in err$(M, P)$) $>$ tol do
4:     //Shift Breg. divergence with min to avoid underflow
5:     E-Step: $Z_{ij} = p_j \exp[-\rho(D_F(X_{i:}||M_{:j})$
6:                 $- \min_j D_F(X_{i:}||M_{:j}))]$
7:         $Z_{ij} = Z_{ij} / \sum_j Z_{ij}$    //normalize $Z$
8:     M-Step: $M = \text{diag}(Z^T \mathbf{1}) Z^T X$
9:         $p = \frac{1}{t} Z^T \mathbf{1}$
10: end while
```

As with exponential family PCA, one can add regularization and kernels to Bregman divergence clustering. Adding instance weights results in a more interesting extension: Bregman normalized cut. Under an identity transfer function (producing a least-squares matching loss), [Xu *et al.*, 2009] illustrated that constrained weighted reverse prediction was equivalent to normalized graph-cut with weights diag$(K\mathbf{1})$. For other transfer functions, we obtain a version of normalized graph-cut generalized to Bregman divergences. Weighting with the kernel matrix intuitively places the most representative points as cluster centers, since the error is high for classifying those points incorrectly; the sum of edge weights across cuts in the adjacency graph, therefore, should be correspondingly low. We expect this extension to have similar properties to normalized cut, such as balanced clustering [Joachims, 2003].

With the generalized reverse prediction framework developed here, connections between existing unsupervised learning algorithms and related gaps have become much more apparent. Similarly, in the next section, we show how the generalized reverse prediction framework can also offer new insights and algorithms for semi-supervised learning.

5 Semi-supervised Learning

We have seen that both supervised and unsupervised learning can be expressed as minimizing a reverse Bregman reconstruction loss. Consequently, as in [Xu *et al.*, 2009], one can perform semi-supervised learning in a unified way using a reverse loss for both labeled and unlabeled data:

$$\min_Z \min_U \; D_{F^*}(Y_L U||\mathbf{f}(X_L))/t_L \;+ \\ \mu \, D_{F^*}(ZU||\mathbf{f}(X_U))/t_U. \qquad (26)$$

Here (X_L, Y_L) and X_U denote the labeled and unlabeled data, t_L and t_U denote the respective number of examples, and the parameter μ trades off between the two losses.

Despite the simplicity of the objective in Equation (26), it has not apparently been investigated in the literature previously. This could be due to the fact that it has been typical to combine a forward loss on the labeled data and a reverse loss (or regularizer) on the unlabeled data [Belkin *et al.*, 2006; Kulis *et al.*, 2009; Zhou *et al.*, 2004]. Given the above discussion, such an approach seems heuristic. A local solution to Equation (26) can be obtained by alternating between optimizing Z and U. We provide pseudocode for semi-supervised clustering and semi-supervised regression in the Appendix.

In the next section, we provide a variance reduction argument showing that more unlabeled data cannot harm the algorithm in a particular sense. In the experiments, we demonstrate the efficacy of the above semi-supervised formulation with a simple alternating minimization.

6 Reverse Loss Decomposition

When using semi-supervised learning, it is important to consider whether adding unlabeled data strictly helps. There is empirical and theoretical evidence that unlabeled data can actually harm the performance of current semi-supervised learning methods [Ben-David *et al.*, 2008; Li and Zhou, 2011; Nadler *et al.*, 2009]. In the following, we demonstrate that the specific semi-supervised approach proposed here should reduce variance in the error estimate, and at worst will not decrease performance over strictly using the labeled data.

Definition 9 (Affine set) *A set Ω is **affine** if for any $x_1, \dots, x_n \in \Omega$ and $a_1, \dots, a_n \in \mathbb{R}$, we have $\sum_{i=1}^n a_i x_i \in \Omega$. Every affine set is convex.*

Definition 10 (Generalized Pythagorean Theorem) *[Murata* et al.*, 2004] For $x_1, x_2 \in \mathcal{X}$ and Ω an affine set, let $P_\Omega(x) = \arg\min_{\omega\in\Omega} D_F(\omega||x)$ be the Bregman projection, then*

$$D_F(x_1||x_2) = D_F(x_1||P_\Omega(x_2)) + D_F(P_\Omega(x_2)||x_2).$$

For reverse semi-supervised learning, we define the subspace based on the current reverse model, U: $\Omega = \{zU \mid zU \in \mathbf{f}(\mathcal{X})\}$, where $\mathcal{X}$ is the space of all possible input vectors, x. For an affine set $\mathcal{Z}$, $\mathcal{Z}U$ is guaranteed to be affine. Examples of transfers that enable $\mathcal{Z}$ to be affine include $\mathbf{f}(x) = x$, $\mathbf{f}(x) = \log_2(x) + 1/\log(2)$ (KL divergence) and the relaxed sigmoid, $\mathbf{f}(x) = \text{sign}(x) \cdot \log(|x|/\theta + 1)$ ($\theta \in \mathbb{R}$). A sufficient thought not necessary condition to make the image affine is for f to be surjective on $\mathbb{R}^k$. In the following theorem, we provide our variance reduction argument for this class of transfer functions.

Theorem 11 *For any X_L, X_U, Y_L, U and transfer function, f, with resulting affine feature set, $\mathcal{Z}$, then for $R(X, YU) = D_{F^*}(YU||f(X))$*

$$\begin{aligned} &E[R(X_L, Y_L U)/t_L] \\ &\quad = \quad E[R(X, Z^* U)/t_S] + E[R(Z_L^* U, Y_L U)/t_L] \quad (27) \end{aligned}$$

where $X = [X_L; X_U]$ *and* $Z^* = \arg\min_{Z \in \mathcal{Z}} D_{F^*}(ZU||f(X))$.

Therefore, with more unlabeled data, the variance of the estimate of the first expected value in the loss decomposition is decreased, monotonically reducing the variance of the supervised learning error estimate. With a reduced variance estimate of the loss, one is closer to optimizing the true loss and should obtain an improved solution.

If we only want to consider convex rather than affine sets, the Generalized Pythagoras Theorem changes to an inequality on the losses

$$D_F(x_1||x_2) \geq D_F(x_1||P_\Omega(x_2)) + D_F(P_\Omega(x_2)||x_2).$$

Unfortunately, this inequality does not provide any guarantees for unlabeled data. The cosine law might enable a similar variance reduction argument if specific properties of the convex set and transfer are known:

$$\begin{aligned} D_F(x_1, x_2) = D_F(x_1, x_3) + D_F(x_3, x_2) \\ - (x_1 - x_3)^T(\nabla F(x_2) - \nabla F(x_3)). \end{aligned}$$

7 Semi-supervised Learning Experiments

In this section, we explore two main points: (1) the utility of non-linear transfers, and (2) the general performance of our principled semi-supervised algorithm. For regression and classification, we generate synthetic data to assess the importance of having the correct transfer function; to test generality, we use real-world data. We report transductive error as some competitors are solely transductive.

Regression Experiments: For regression, we generate synthetic data with three transfer functions: (i) $Y = XW$, (ii) $Y = (XW)^3$ and (iii) $Y = \exp(XW)$. The data is generated in reverse with Y and U generated from $(0, 1)$-Gaussians and X set to $X = f^{-1}(YU+\text{noise})$. We also report results on three UCI datasets: kin-32fh (n=34, k = 1), puma-8nm (n = 8, k =1) and California housing (n = 5, k =1). We compare our approach against transductive regression [Cortes and Mohri, 2007] and also include supervised (kernel) least-squares as a baseline comparison. We use limited memory BFGS to optimize our objectives.

To better evaluate the algorithms, we automatically tune parameters using transductive error on the unlabeled data for each algorithm on each dataset. In practice, parameters are usually tuned using cross-validation on only labeled examples; this approach, however, can have confounding effects due to lack of labeled data for evaluation. Since cross-validation on labeled examples is a high-variance estimate of the true performance of different parameter settings, using the transductive error enables us to more accurately select parameters for each algorithm on each dataset. For both regression and classification, we included cross validation results on one real dataset (the starred dataset) to illustrate that cross validation can be used in practice. We tuned the trade-off parameter $\mu \in$ [1e-3, 1e-2, 1e-1] (above $\mu = 1$, performance degrades). For transductive regression, we tuned over λ, C_1 and C_2 and fixed r as recommended in their paper. All algorithms were tuned over using no kernel, a linear kernel and Gaussian kernels with widths in $[0.01,\ 0.1,\ 1,\ 5,\ 10]$.

The results in Table 1 clearly indicate that using the correct transfer function is crucial to performance. For each of the synthetic datasets, optimizing with the transfer used to generate the synthetic data performs significantly better than the other algorithms and objectives, verifying our expectations. The synthetic results for the exponential transfer are particularly illustrative: errors are drastically amplified for non-exponential data (the wrong transfer can significantly impede prediction), but much lower for exponential data.

These insights and properties transfer to performance on real datasets. On the kin-32fh dataset, using an exponential transfer considerably improved performance. This surprising improvement likely occurred because the kin-32fh outputs were all positive and the exponential transfer enforces $Y_{ij} \in \mathbb{R}^+$. On the mainly linear simulated robot-arm dataset, puma-8nm, the three linear approaches are comparable. The highly nonlinear California housing dataset, however, illustrates some interesting phenomena. First, the addition of kernels is important for modeling: the transductive regression algorithm leverages the wide range of kernels well for modeling (as reducing the number of widths causes it's performance to degrade below supervised learning). Second, the nonlinear cube transfer interestingly performed the best out of the three transfers, illustrating that adding nonlinear transfers to the reverse prediction framework empirically as well as theoretically increases the breadth of applicable datasets.

Classification Experiments: For classification, we generated synthetic data with three transfer functions: (i) $Y = XW$, (ii) $Y = (1 + \exp(-XW))^{-1}$ (sigmoid) and (iii) $Y = \exp(XW)(\mathbf{1}^T \exp(XW))^{-1}$ (softmax). The data was generated by selecting uniformly random classes for Y and drawing U from a Gaussian distribution for the identity transfer and from a uniform distribution for sigmoid and softmax, with rows of U and the noise normalized for softmax. For sigmoid and softmax, $X = f^{-1}((1-\sigma)YU + \sigma{\cdot}\text{noise})$. We also tested on three real datasets: the Wisconsin breast cancer (WBC) dataset with (n=10, k=2), LINK, a WebKB dataset with (n=1051, k=2), and SetStr, a semi-supervised benchmark dataset with (n = 15, k=2).

Table 1: Average transductive error of semi-supervised regression techniques on a variety of datasets, with (n, k, t_u) and $t_l = 20$, over 5 splits of the data. Parameters were tuned with cross-validation on the Parkinson's dataset (starred).

	SYN-GAUSS	SYN-CUBED	SYN-EXP	KIN-32FH	PUMA-8NM	CALHOUSING*
	N=30, K=5, U=200	N=20, K=3, U=200	N=5, K=2, U=200	N=34, K=1, U=100	N=8, K=1, U=100	N=5, K=1, U=300, t_l=50
SUP-KERNEL	**2e-14 ± 4.4e-15**	0.246 ± 0.030	408 ± 116.9	0.759 ± 0.249	**27.36 ± 1.858**	133.3 ± 10.01
TRANS-REG	3E-06 ± 2E-06	0.244 ± 0.0672	1039 ± 304.7	0.766 ± 0.254	**27.04 ± 3.270**	**119.6 ± 10.64**
TRANS-REG	3E-06 ± 8.9E-07	0.244 ± 0.030	1039 ± 136.2	0.766 ± 0.113	**27.04 ± 1.462**	**119.6 ± 4.758**
SEMI EUC	**6e-31 ± 8.9e-32**	0.120 ± 0.003	1423 ± 95.61	0.524 ± 0.042	**28.91 ± 1.420**	129.5 ± 7.101
SEMI CUBED	1.91 ± 0.752	**3e-05 ± 2.6e-06**	348.1 ± 34.03	2.19 ± 0.152	4300 ± 2733.	**127.1 ± 6.064**
SEMI EXP	2E+18 ± 1.341	5E+83 ± 4.024	**1.7e-4 ± 0.000**	**0.222 ± 0.014**	3E+18 ± 1.788	138.5 ± 4.006

Table 2: Average transductive percent misclassification error of semi-supervised classification techniques on synthetic and real-world datasets, given (n, k, t_u) and $t_l = 10$, over 20 splits. Euc, Sig and Soft correspond to objectives with identity, sigmoid and softmax transfers. Hard (Soft) corresponds to hard (soft) clustering. Hard/Soft Sig NC is Bregman normalized cut with a sigmoid transfer. Parameters were tuned with cross-validation on the SetStr dataset (starred) with $t_l = 50$.

	SYN-GAUSS	SYN-SIGMOID	SYN-SOFTMAX	WBC	LINK	SETSTR*
	N=30, K = 5, U=100	N=10, K=3, U=100	N=10, K = 3, U=100	N=10, K = 2, U=50	N=1051, K=2, U=200	N=15, K=2, U=400
SUP-KERNEL	3.70 ± 0.450	27.7 ± 1.621	29.9 ± 4.136	11.2 ± 1.283	0.207 ± 0.029	0.501 ± 0.007
LGC	0.90 ± 0.205	22.6 ± 1.870	26.5 ± 2.423	5.20 ± 0.963	0.150 ± 0.008	0.476 ± 0.014
LAPSVM	**0 ± 0.0**	17.2 ± 1.443	24.8 ± 3.785	4.80 ± 0.606	0.136 ± 0.011	0.479 ± 0.009
LAPRLSC	**0 ± 0.0**	18.2 ± 1.514	26.5 ± 3.781	3.60 ± 0.521	0.144 ± 0.016	0.504 ± 0.012
HARD EUC	**0 ± 0.0**	14.3 ± 0.991	27.4 ± 3.537	**3.20 ± 0.218**	0.130 ± 0.007	0.481 ± 0.019
HARD SIG	2.70 ± 0.296	16.2 ± 0.829	27.1 ± 3.532	**3.20 ± 0.218**	0.130 ± 0.007	**0.445 ± 0.015**
HARD SFTMAX	3.40 ± 0.268	13.7 ± 0.554	**22.2 ± 4.241**	6.00 ± 0.282	0.134 ± 0.007	0.503 ± 0.017
HARD SIG NC	2.70 ± 0.296	16.2 ± 0.829	27.1 ± 3.532	6.80 ± 0.606	0.130 ± 0.007	0.510 ± 0.018
SOFT EUC	**0 ± 0.0**	12.5 ± 0.803	35.0 ± 4.387	4.40 ± 0.334	**0.105 ± 0.007**	0.489 ± 0.013
SOFT SIG	**0 ± 0.0**	**12.0 ± 0.902**	32.7 ± 4.380	4.40 ± 0.521	0.193 ± 0.030	0.481 ± 0.019
SOFT SFTMAX	1.90 ± 0.584	13.2 ± 0.944	**19.0 ± 3.117**	5.20 ± 0.456	0.231 ± 0.027	0.481 ± 0.019
SOFT SIG NC	**0 ± 0.0**	**11.6 ± 0.920**	33.3 ± 4.637	**4.00 ± 0.400**	0.114 ± 0.006	0.480 ± 0.014

We compare to three state-of-the-art semi-supervised algorithms: learning with local and global consistency (LGC) [Zhou *et al.*, 2004], Laplacian SVMs [Sindhwani *et al.*, 2005] and Laplacian regularized least squares (RLSC) [Sindhwani *et al.*, 2005]. We test our Bregman hard and soft clustering objectives over a variety of transfer functions, kernels and instance weights. We tuned the trade-off parameter, $\mu \in$ [1e-3, 1e-2, 1e-1] and soft clustering parameter, $\rho \in [1,\ 10,\ 50,\ 100,\ 200]$. For LGC, we tuned the smoothness parameter $\alpha \in$ [1e-5, 1e-3, 0.1, 0.5]. For the Laplacian algorithms, we set $\gamma_A =$ 1e-6 and $\gamma_I = 0.01$ and tuned the Laplacian degree in $[1,\ 2,\ 5]$ and number of nearest neighbours in $[5,\ 20]$. All algorithms were tuned over no kernel, a linear kernel and a Gaussian kernel with widths in $[0.01,\ 0.1,\ 1,\ 5,\ 10]$.

As with regression, the transfer has an impact on performance (Table 2). Though all algorithms performed well on the Gaussian data, we can see that the incorrect sigmoidal and softmax transfers did decrease performance. We see a corresponding result for the synthetic sigmoidal and softmax data. Interestingly, for the sigmoidal transfer and for the WBC dataset, the normalized cut extension improved performance. Moreover, it seems to be the case that the normalized cut extension performs better with the soft clustering algorithm rather than the hard clustering algorithm. On the three real-world datasets, it is interesting that there is no obvious trend of hard over soft clustering or one transfer over another. On WBC, hard clustering with a euclidean (linear) or sigmoidal transfer performs the best; on LINK, soft clustering with a euclidean transfer performs the best; and on SetStr, hard clustering with a sigmoidal transfer performs the best. Overall, we can see that the variety of options provided by the reverse prediction framework enables us to tailor our objective to the given data. Using prior knowledge alongside empirical selection of transfers, prediction accuracy can be dramatically improved.

8 Conclusion

In this paper, we demonstrated that reframing supervised and unsupervised learning as reverse prediction permits key algorithmic insights and empirical benefits. With one learning paradigm, unsupervised objectives can be designed to obtain certain properties by varying the constraints on the unknown labels and using different transfer functions, kernels, regularizers and instance weights. The resulting principled semi-supervised algorithm outperforms state-of-the-art, particularly when Gaussian assumptions are not met.

There are still many open questions about how to further generalize the reverse prediction framework. One useful extension would be to relax the restrictions on the transfer function: there are several useful nonsmooth, noninvertible transfers, like the sign transfer for hinge loss. Another avenue is to systematically determine which other unsupervised algorithms are instances of Bregman reverse prediction, with potential novel insights for generalization.

References

A Argyriou, C Micchelli, and M Pontil. When is there a representer theorem? Vector versus matrix regularizers. *JMLR*, 10:2507–2529, 2009.

P Auer, M Herbster, and MK Warmuth. Exponentially many local minima for single neurons. *Advances in Neural Information Processing Systems*, 1996.

M-F. Balcan and A. Blum. A PAC-style model for learning from labeled and unlabeled data. In *Annual Conference on Computational Learning Theory (COLT)*, 2005.

A Banerjee, S Merugu, IS Dhillon, and J Ghosh. Clustering with bregman divergences. *The Journal of Machine Learning Research*, 6:1705–1749, 2005.

M Belkin, P Niyogi, and V Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *JMLR*, 2006.

S Ben-David, T Lu, and David Pal. Does unlabeled data provably help? worst-case analysis of the sample complexity of semi-supervised learning. *Annual Conference on Computational Learning Theory*, pages 33–44, 2008.

T De Bie and Nello Cristianini. Convex methods for transduction. *Advances in Neural Information Processing Systems*, 16:73–80, 2003.

CM Bishop. Pattern recognition and machine learning. *Springer*, 2006.

A Blum and T Mitchell. Combining labeled and unlabeled data with co-training. *Annual Conference on Computational Learning Theory*, pages 92–100, 1998.

S Canu and A Smola. Kernel methods and the exponential family. *Neurocomputing*, 2006.

M Collins, S Dasgupta, and R E Schapire. A generalization of principal component analysis to the exponential family. *Advances in Neural Information Processing Systems*, 2002.

A Corduneanu and T Jaakkola. Data dependent regularization. *Semi-Supervised Learning*, 2006.

C Cortes and M Mohri. On transductive regression. *Advances in Neural Information Processing Systems*, 19:305, 2007.

G Druck and A McCallum. High-performance semi-supervised learning using discriminatively constrained generative models. *International Conference on Machine Learning*, 2010.

Gregory Druck and Andrew McCallum. High-performance semi-supervised learning using discriminatively constrained generative models. In *International Conference on Machine Learning*, 2010.

Gregory Druck, Chris Pal, Xiaojin Zhu, and Andrew McCallum. Semi-supervised classification with hybrid generative/discriminative methods. In *International Conference on Knowledge Discovery and Data Mining*, 2007.

W. Ehm, M.G. Genton, and T. Gneiting. Stationary covariances associated with exponentially convex functions. *Bernoulli*, 9(4):607–615, 2003.

C Gentile and M Warmuth. Linear hinge loss and average margin. *Advances in Neural Information Processing Systems*, 1999.

G Gordon. Generalized^2 linear^2 models. *Advances in Neural Information Processing Systems*, Jan 2003.

DP Helmbold, J Kivinen, and MK Warmuth. Relative loss bounds for single neurons. *IEEE Transactions on Neural Networks*, 10(6):1291–1304, 2002.

T Joachims. Transductive inference for text classification using support vector machines. *International Conference on Machine Learning*, 1999.

T Joachims. Transductive learning via spectral graph partitioning. *International Conference on Machine Learning*, pages 290–297, 2003.

G. Kimeldorf and G. Wahba. Some results on tchebycheffian spline functions. *Journal of Mathematical Analysis and Applications*, 33(1):82–95, 1971.

J Kivinen and MK Warmuth. Relative loss bounds for multidimensional regression problems. *Journal of Machine Learning Research*, 45(3):301–329, 2001.

B Kulis, S Basu, I Dhillon, and R Mooney. Semi-supervised graph clustering: a kernel approach. *Journal of Machine Learning Research*, 74:1–22, 2009.

Julia A. Lasserre, Christopher M. Bishop, and Thomas P. Minka. Principled hybrids of generative and discriminative models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

Y Li and Z Zhou. Towards making unlabeled data never hurt. *International Conference on Machine Learning*, 2011.

N Murata, T Takenouchi, T Kanamori, and S Eguchi. Information geometry of u-boost and bregman divergence. *Neural Computation*, 16(7):1437–1481, 2004.

B Nadler, N Srebro, and X Zhou. Semi-supervised learning with the graph laplacian: The limit of infinite unlabelled data. *Advances in Neural Information Processing Systems*, 2009.

K Nigam, A McCallum, S Thrun, and T Mitchell. Text classification from labeled and unlabeled documents using em. *Journal of Machine Learning Research*, 39:103–134, 2000.

V Sindhwani, P Niyogi, and M Belkin. Beyond the point cloud: from transductive to semi-supervised learning. *International Conference on Machine Learning*, 2005.

L Xu, M White, and D Schuurmans. Optimal reverse prediction: a unified perspective on supervised, unsupervised and semi-supervised learning. *International Conference on Machine Learning*, 2009.

D Zhou, O Bousquet, TN Lal, J Weston, and B Scholkopf. Learning with local and global consistency. *Advances in Neural Information Processing Systems*, 2004.

X Zhu. Semi-supervised learning literature survey. *Computer Science*, 2006.

9 Appendix

9.1 Proofs

Lemma 4 $D_F(\hat{\mathbf{z}}||\mathbf{z}) = D_{F^*}(\mathbf{y}||\hat{\mathbf{y}})$ *where* $\hat{\mathbf{y}} = \mathbf{f}(\hat{\mathbf{z}})$ *and* $\mathbf{y} = \mathbf{f}(\mathbf{z})$.

Proof: Recall that $F^*(\mathbf{y}) = \max_{\mathbf{z}} \mathbf{z}^T\mathbf{y} - F(\mathbf{z})$, and solving for the maximum $\mathbf{z}$ we obtain

$$\begin{aligned} \frac{d}{d\mathbf{z}} &= \mathbf{y} - \nabla F(\mathbf{z}) = \mathbf{y} - \mathbf{f}(\mathbf{z}) = 0 \\ &\implies \mathbf{z} = \mathbf{f}^{-1}(\mathbf{y}) \end{aligned}$$

giving

$$F^*(\mathbf{y}) = \mathbf{f}^{-1}(\mathbf{y})^T\mathbf{y} - F(\mathbf{f}^{-1}(\mathbf{y}))$$

Now we can rewrite $D_{F^*}(\mathbf{y}||\hat{\mathbf{y}})$ in terms of F and $\mathbf{f}^{-1} = \mathbf{f}^*$

$$\begin{aligned} D_{F^*}(\mathbf{y}||\hat{\mathbf{y}}) &= F^*(\mathbf{y}) - F^*(\hat{\mathbf{y}}) - \mathbf{f}^*(\hat{\mathbf{y}})^T(\mathbf{y} - \hat{\mathbf{y}}) \\ &= \mathbf{f}^{-1}(\mathbf{y})^T\mathbf{y} - F(\mathbf{f}^{-1}(\mathbf{y})) - \mathbf{f}^{-1}(\hat{\mathbf{y}})^T\hat{\mathbf{y}} + F(\mathbf{f}^{-1}(\hat{\mathbf{y}})) - \mathbf{f}^{-1}(\hat{\mathbf{y}})^T(\mathbf{y} - \hat{\mathbf{y}}) \\ &= \mathbf{f}^{-1}(\mathbf{y})^T\mathbf{y} - F(\mathbf{f}^{-1}(\mathbf{y})) + F(\mathbf{f}^{-1}(\hat{\mathbf{y}})) - \mathbf{f}^{-1}(\hat{\mathbf{y}})^T\mathbf{y} \end{aligned}$$

Finally, recall that $\hat{\mathbf{y}} = \mathbf{f}(\hat{\mathbf{z}})$ and $\mathbf{y} = \mathbf{f}(\mathbf{z})$, giving us

$$\begin{aligned} D_{F^*}(\mathbf{y}||\hat{\mathbf{y}}) &= \\ &= \mathbf{f}^{-1}(\mathbf{y})^T\mathbf{y} - F(\mathbf{f}^{-1}(\mathbf{y})) + F(\mathbf{f}^{-1}(\hat{\mathbf{y}})) - \mathbf{f}^{-1}(\hat{\mathbf{y}})^T\mathbf{y} \\ &= \mathbf{f}^{-1}(\mathbf{f}(\mathbf{z}))^T\mathbf{f}(\mathbf{z}) - F(\mathbf{f}^{-1}(\mathbf{f}(\mathbf{z}))) + F(\mathbf{f}^{-1}(\mathbf{f}(\hat{\mathbf{z}}))) - \mathbf{f}^{-1}(\mathbf{f}(\hat{\mathbf{z}}))^T\mathbf{f}(\mathbf{z}) \\ &= \mathbf{z}^T\mathbf{f}(\mathbf{z}) - F(\mathbf{z}) + F(\hat{\mathbf{z}}) - \hat{\mathbf{z}}^T\mathbf{f}(\mathbf{z}) \\ &= F(\hat{\mathbf{z}}) - F(\mathbf{z}) - \mathbf{f}(\mathbf{z})^T(\hat{\mathbf{z}} - \mathbf{z}) \\ &= D_F(\hat{\mathbf{z}}||\mathbf{z}) \end{aligned}$$

■

Theorem 1 *Given rank n input $t \times n$ matrix, X, and rank k output $t \times k$ matrix, Y, with $t > n$ and $t > k$, there exist unique global minimizers, W^* and U^* for $L(XW, Y)$ and $R(X, YU)$ respectively:*

$$W^* = \operatorname*{argmin}_W D_F(XW||f^{-1}(Y)) \tag{28}$$

$$U^* = \operatorname*{argmin}_U D_{F^*}(YU||f(X)) \tag{29}$$

Moreover, W^* and U^* are related in the following way

$$X^T\mathbf{f}(XW^*) = \mathbf{f}^{-1}(YU^*)^TY \tag{30}$$

Proof: Let F be a strictly convex function with Dom$(F) = \{XW : W \in \mathbb{R}^n\}$, with any full rank X (i.e., X such that $XW_1 \neq XW_2$ for $W_1 \neq W_2$). Then $G = F(X\cdot)$ has Dom$(G) = \mathbb{R}^n$ (which is convex). For W_1, W_2 in Dom(G) such that $W_1 \neq W_2$

$$\begin{aligned} G(\lambda W_1 + (1-\lambda)W_2) &= F(X(\lambda W_1 + (1-\lambda)W_2)) \\ &= F(\lambda XW_1 + (1-\lambda)XW_2) \\ &< \lambda F(XW_1) + (1-\lambda)F(XW_2) \quad \text{because } F \text{ is strictly convex and } XW_1 \neq XW_2. \\ &= \lambda G(W_1) + (1-\lambda)G(W_2) \end{aligned}$$

Therefore, G is strictly convex. The optimization $\min_W G(W)$ therefore has a unique minimum. Notice that we can always linearize X, W and Y to make sure that we are working with vectors.

For the relation, since W^* and U^* are global minimizers of $L(XW, Y)$ and $R(X, YU)$, we know that the gradients

$$\begin{aligned} \frac{d}{dW} L(XW^*, Y) &= X^T \left(\mathbf{f}(XW^*) - Y\right) = \mathbf{0} \quad (n \times k) \\ \frac{d}{dU} R(X, YU^*) &= Y^T \left(\mathbf{f}^*(YU^*) - X\right) = \mathbf{0} \quad (k \times n) \end{aligned}$$

giving

$$\begin{aligned} X^T \left(\mathbf{f}(XW^*) - Y\right) &= (Y^T \left(\mathbf{f}^*(YU^*) - X\right))^T \\ X^T \mathbf{f}(XW^*) - X^T Y &= \mathbf{f}^*(YU)^T Y - X^T Y \end{aligned}$$

$\Longrightarrow$

$$X^T \mathbf{f}(XW^*) = \mathbf{f}^*(YU^*)^T Y$$

■

Theorem 11 *For any X_L, X_U, Y_L, U and transfer function, f, with resulting affine feature set, $\mathcal{Z}$, then for $R(X, YU) = D_{F^*}(YU||f(X))$*

$$\begin{aligned} E[R(X_L, Y_L U)/t_L] = E[R(X, Z^* U)/t_S] + \\ E[R(Z_L^* U, Y_L U)/t_L] \end{aligned} \tag{31}$$

where $X = [X_L; X_U]$ and $Z^* = \operatorname*{argmin}_{Z \in \mathcal{Z}} D_{F^*}(ZU||f(X))$

Proof: From the Generalized Pythagoras Theorem, we know that

$$\begin{aligned} &E[D_{F^*}(Y_L U||f(X_L))/t_L] = \\ &\quad E[D_{F^*}(Z_L^* U||f(X_L))/t_L] + E[D_{F^*}(Z_L^* U||Y_L U)/t_L] \end{aligned}$$

Since

$$\begin{aligned} E[D_{F^*}(Z_L^* U||f(X_L))/t_L] &= E[D_{F^*}(Z^* U||f(X))/t_S] \\ &= E[R(X, Z^* U)/t_S] \end{aligned}$$

we get the above result. ■

9.2 Algorithms for clustering

To obtain the simplifications used for our modified clustering algorithms, we provide the following lemmas.

Lemma 12 $D_{F^*}(YU||\mathbf{f}(X)) = D_F(X||\mathbf{f}^*(YU)) = D_F(X||Y\mathbf{f}^*(U))$

Proof: From Lemma 4, we know that $D_F(X||\mathbf{f}^*(YU)) = D_{F^*}(YU||\mathbf{f}(X))$. Now, since $Y \in \{0,1\}^{t \times k}$ and $Y\mathbf{1} = \mathbf{1}$, we can see that YU simply selects rows of U, i.e. if there is a one at position $1 \leq j \leq k$, then row j in U is selected. Therefore,

$$\mathbf{f}^*(YU) = Y\mathbf{f}^*(U)$$

and we conclude that $D_F(X||\mathbf{f}^*(YU)) = D_F(X||Y\mathbf{f}^*(U))$. We can now optimize over M for $D_F(X||YM)$. ■

Lemma 13 *For a given $Y \in \{0,1\}^{t \times k}$ with $Y\mathbf{1} = \mathbf{1}$ and class j with $X \in Dom(\mathbf{f})$,*

$$\frac{1}{\mathbf{1}^T Y_{:j}} \sum_{i:Y_{ij}=1} X_{i:} = \operatorname*{argmin}_{M \in Dom(\mathbf{f})} \sum_{i:Y_{ij}=1} D_F(X_{i:}||M_{:j})$$

Proof: Let n_j be the number of instances with class j, $\mathbf{m} = M_{:j}$, $\bar{\mathbf{x}} = \frac{1}{n_j}\sum_{i:Y_{ij}=1} X_{i:}$, $(\mathbf{m}, \bar{\mathbf{x}} \in \mathbb{R}^{n\times 1})$ and $\bar{F} = \frac{1}{n_j}\sum_{i:Y_{ij}=1} F(X_{i:})$. Now to simply $\frac{1}{n_j}\sum_{i:Y_{ij}=1} D_F(X_{i:}||\mathbf{m})$

$$\begin{aligned}\frac{1}{n_j}\sum_{i:Y_{ij}=1} D_F(X_{i:}||\mathbf{m}) &= \frac{1}{n_j}\sum_{i:Y_{ij}=1} F(X_{i:}) - F(\mathbf{m}) - \mathbf{f}(\mathbf{m})^T(X_{i:} - \mathbf{m})\\ &= \bar{F} - \frac{1}{n_j}\sum_{i:Y_{ij}=1} F(\mathbf{m}) - \mathbf{f}(\mathbf{m})^T \frac{1}{n_j}\sum_{i:Y_{ij}=1}(X_{i:} - \mathbf{m})\\ &= \bar{F} - F(\mathbf{m}) - \mathbf{f}(\mathbf{m})^T(\bar{\mathbf{x}} - \mathbf{m})\end{aligned}$$

and by definition

$$D_F(\bar{\mathbf{x}}||\mathbf{m}) = F(\bar{\mathbf{x}}) - F(\mathbf{m}) - \mathbf{f}(\mathbf{m})^T(\bar{\mathbf{x}} - \mathbf{m})$$

$$\Longrightarrow$$

$$\frac{1}{n_j}\sum_{i:Y_{ij}=1} D_F(X_{i:}||\mathbf{m}) = \bar{F} - F(\bar{\mathbf{x}}) + D_F(\bar{\mathbf{x}}||\mathbf{m})$$

$$\Longrightarrow$$

$$\sum_{i:Y_{ij}=1} D_F(X_{i:}||\mathbf{m}) = n_j\bar{F} - n_jF(\bar{\mathbf{x}}) + n_jD_F(\bar{\mathbf{x}}||\mathbf{m})$$

$$\Longrightarrow$$

$$\min_{\mathbf{m}}\sum_{i:Y_{ij}=1} D_F(X_{i:}||\mathbf{m}) = \min_{\mathbf{m}} D_F(\bar{\mathbf{x}}||\mathbf{m})$$

Since the Bregman divergence is guaranteed to be greater than or equal to zero, the minimum value for $D_F(\bar{\mathbf{x}}||\mathbf{m})$ is zero, obtained by setting $\mathbf{m} = \bar{\mathbf{x}}$. Therefore, for each instance i, the optimal setting for the inner minimization of $M_{:j} = \frac{1}{n_j}\sum_{i:Y_{ij}=1} X_{i:}$.

Notice that the objective value therefore is $n_j\bar{F} - n_jF(\bar{x})$, which is always non-negative because F is strictly convex so:

$$F(\bar{\mathbf{x}}) = F\left(\sum_{i:Y_{ij}=1}\frac{X_{i:}}{n_j}\right) < \sum_{i:Y_{ij}=1}\frac{1}{n_j}F(X_{i:}) = \bar{F}$$

■

From Lemmas 12 and 13, we get the following simplifications for Bregman hard clustering with non-linear transfers (Equations (22) and (24) in the main paper).

$$\begin{aligned}&\min_{Z\in\mathcal{Z}}\min_{U\in Dom(f^*)} D_{F^*}(ZU||f(X)) =\\ &\quad= \min_{Z\in\mathcal{Z}}\min_{M\in Dom(f)\subset\mathbb{R}^{n\times k}} D_{F^*}(X||ZM)\\ &\quad= \min_{Z\in\mathcal{Z}}\min_{M\in Dom(f)}\sum_{j=1}^{k}\sum_{i:Z_{ij}=1} D_{F^*}(X_{i:}||YM_{:j})\\ &\quad= \min_{Z\in\mathcal{Z}}\sum_{j=1}^{k}\frac{1}{\mathbf{1}^TZ_{:j}}\sum_{i:Z_{ij}} X_{i:}\end{aligned}$$

For mixture model clustering using standard EM, the unsimplified optimization given by Banerjee et al. [2005], with $\mathcal{S} = \{Z \in [0,1] \mid Z\mathbf{1} = 1\}$

$$\min_{Z\in\mathcal{S}}\min_{U}\sum_{i=1}^{t}\sum_{j=1}^{k} -\log(P_{F^*}(X_i|U_j))Z_{ij} \tag{32}$$

$$= \min_{Z\in\mathcal{S}}\min_{U\in\text{Dom}(f)}\sum_{i=1}^{t}\sum_{j=1}^{k} -\log\left(e^{-D_F(X_i||U_j)}\right)Z_{ij} \tag{33}$$

They simplify the M-step using similar arguments to those for hard clustering. We define a slightly different optimization, now optimizing for the transfer $M = f(U)$ and then illustrate that we simplify the M-step for non-linear transfers. Note that in our optimization we move the sum and probability scaling inside the $\log$; this does not change the optimum because $\log$ is monotonic and the probabilities are always greater than or equal to zero. Note that we also add a smoothness parameter ρ; as$\rho \to \infty$, the objective approaches the hard clustering objective.

$$\min_{Z \in \mathcal{S}} \min_{M \in \text{Dom}(f)} -\sum_{i=1}^{t} \log \left(\sum_{j=1}^{k} e^{-\rho D_F(X_i || M_j)} Z_{ij} \right) = \tag{34}$$

$$= \min_{\mathbf{p} \geq \mathbf{0}, \mathbf{p}^T \mathbf{1} = 1} \min_{M \in \text{Dom}(f)} -\sum_{i=1}^{n} \log \left(\sum_{j=1}^{k} p_j e^{-\rho D_F(X_i || M_{j:})} \right) \tag{35}$$

Again, the inner minimization over M simplifies to an expectation

$$M_{:j} = \frac{1}{\mathbf{1}^T Y_{:j}} \sum_{i=1}^{n} Y_{ij} X_{i:} \tag{36}$$

and we get the updates shown in Algorithm 2. For more details, look at the simplifications in Banerjee et al [2005].

9.3 Pseudocode and transfer functions

Below we provide pseudocode for our semisupervised regression approach, in Algorithm 9.3, and our semisupervised classification approach, Algorithm 9.3. The classification algorithm uses similar tricks from the unsupervised clustering algorithms provided in the paper. The regression algorithm simply uses a smooth optimizer (like limited memory BFGS) to alternate between optimizing Z and U according to the objective provided in Equation 26.

Algorithm 3 RevSemiSupRegression($X_L, X_U, Y_L, D_F, \boldsymbol{\beta}$)

```
1: // $\boldsymbol{\beta}$ is a weighting on samples, e.g. $\boldsymbol{\beta} = [\mathbf{1}; \boldsymbol{\mu}]$
2: Initialize $Y_U$ and $U$
3: $X = [X_L; X_U],\ K = k(X, X),\ \alpha = 0.1$
4: err($Y_U, U$) = $\beta D_{F^*}([Y_L; Y_U] || \mathbf{f}(K))$
5: while (change in err($Y_U, U$)) $>$ tol do
6:     $U = \text{argmin}_U\, err(Y_U, U)$
7:     $Y_U = \text{argmin}_Z\, err(Z, U)$
8: end while
9: $Y = [Y_L; Y_U]$
10: $A^* = \text{argmin}_A\, D_F(KA || f^{-1}(Y)) + \alpha \text{tr}(AA^T K)$
```

Below are the potential functions, inverses, forward losses and reverse losses for the transfers used in the paper. To make the tables cleaner, sometimes we will refer to $\hat{\mathbf{x}} = \mathbf{f}^{-1}(\mathbf{y}U)$. We omit D_{F^*} because it is not used in the clustering algorithm and is long, making the table difficult to read.

Algorithm 4 RevSemiSupSoftCluster($X_L, X_U, Y_L, D_F, \boldsymbol{\beta}$)

```
1:  Initialize $M$ (e.g. $k$ randomly selected rows from $X$)
2:  $p = \mathbf{1}/k$
3:  $Y_U = [\,]$
4:  $X = [X_L; X_U]$
5:  err$(M, p) = -\sum_i \log\left(\sum_j p_j \exp(-\rho\boldsymbol{\beta} D_F(X_{i:}||M_{:j}))\right)$
6:  **while** (change in err$(M, p)$) $>$ tol **do**
7:      //Shift Breg. divergence with min to avoid underflow
8:      **E-Step:**
9:              $Y_U(i,j) = p_j \exp[-\rho\mu(D_F(X_U(i,:)||M_{:j}) - \min_j D_F(X_U(i,:)||M_{:j}))]$
10:             $Y_U(i,j) = Y_U(i,j)/\sum_j Y_U(i,j)$
11:             $Z = [Y_L; Y_U]$
12:     **M-Step:**
13:             $M = \text{diag}(Z^T\mathbf{1})Z^T X$
14:             $p = \frac{1}{t} Z^T \mathbf{1}$
15: **end while**
```

Table 3: Transfer functions with their inverses and potential functions.

	$f(x)$	$f^{-1}(y)$	$F(\mathbf{x})$	$F^*(\mathbf{y})$
IDENTITY	$\mathbf{x}$	$\mathbf{x}$	$\mathbf{x}^2/2$	$\mathbf{y}^2/2$
SIGMOID	$\sigma(x) = (1+e^{-\mathbf{x}})^{-1}$	$\ln(\mathbf{y}/(\mathbf{1}-\mathbf{y}))$	$\mathbf{1}^T\ln(\mathbf{1}+e^{-\mathbf{x}})$	$\mathbf{y}\ln(\mathbf{y}/(\mathbf{1}-\mathbf{y})) + \mathbf{1}\ln(\mathbf{1}-\mathbf{y})$
SOFTMAX	$\xi(x) = e^{\mathbf{x}}/\mathbf{1}^T e^{\mathbf{x}}$	$\ln(\mathbf{y}) - \ln(\mathbf{y}_k)\mathbf{1}$	$\ln(\mathbf{1}^T e^{\mathbf{x}})$	$[\ln(\mathbf{y}) - \ln(\mathbf{y}_k)\mathbf{1}]\mathbf{y} - \ln(\mathbf{1}^T(\mathbf{y} - \mathbf{y}_k\mathbf{1}))$
EXP	$e^{\mathbf{x}}$	$\ln(\mathbf{y})$	$\mathbf{1}^T e^{\mathbf{x}}$	$[\ln(\mathbf{y}) - \mathbf{1}]\mathbf{y}^T$
CUBE	$\mathbf{x}^3$	$\mathbf{x}^{1/3}$	$\mathbf{1}^T\mathbf{x}^4/4$	$\mathbf{y}^{1/3}\mathbf{y}^T - 0.25\mathbf{y}^{4/3}\mathbf{1}$

Table 4: Transfer functions with forward and reverse losses.

	$D_F(\mathbf{x}W\|\|f^*(\mathbf{y}))$	$D_{F^*}(\mathbf{y}U\|\|f(\mathbf{x}))$
IDENTITY	$(\mathbf{x}W - \mathbf{y})^2/2$	$(\mathbf{x} - \mathbf{y}U)^2/2$
SIGMOID	$\mathbf{y}\ln(\mathbf{y}/\sigma(\mathbf{x}W)) + (\mathbf{1}-\mathbf{y})\ln((\mathbf{1}-\mathbf{y})/(\mathbf{1}-\sigma(\mathbf{x}W)$	$\mathbf{x}((\mathbf{1} - e^{-\mathbf{x}})/(\mathbf{1} + e^{-\mathbf{x}}))^T - \hat{\mathbf{x}}((\mathbf{1} - e^{-\hat{\mathbf{x}}})/(\mathbf{1} + e^{-\hat{\mathbf{x}}}))^T + \mathbf{1}\ln((\mathbf{1} + e^{-\hat{\mathbf{x}}})/(\mathbf{1} + e^{-\mathbf{x}}))^T$
SOFTMAX	$\ln(e^{\mathbf{x}W}\mathbf{1}^T) - \ln(\mathbf{1}(\mathbf{y} - \mathbf{y}_k\mathbf{1})^T) - \mathbf{y}W^T\mathbf{x}^T + \mathbf{y}(\mathbf{y} - \mathbf{y}_k\mathbf{1})^T$	OMITTED
EXP	$\mathbf{1}^T e^{\mathbf{x}W} - \mathbf{y}W^T\mathbf{x}^T$	$[\ln(\mathbf{y}U) - \mathbf{x} - \mathbf{1}]U^T\mathbf{y}^T + e^{\mathbf{x}}$
CUBE	$((\mathbf{x}W)^4\mathbf{1}^T)/4 - \mathbf{y}W^T\mathbf{x}^T$	$(\mathbf{y}U)^{1/3}U^T\mathbf{y}^T - 0.25(\mathbf{y}U)^{4/3}\mathbf{1} - \mathbf{x}U^T\mathbf{y}^T$