Deep Low-Density Separation for Semi-Supervised Classification⋆

Michael C. Burkhart[1][0000−0002−2772−5840] and Kyle Shan[2][0000−0003−4679−4099]

[1] Adobe Inc., San José, USA
`mburkhar@adobe.com`
[2] Stanford University, Stanford, USA
`kylecshan@gmail.com`

Abstract Given a small set of labeled data and a large set of unlabeled data, semi-supervised learning (SSL) attempts to leverage the location of the unlabeled datapoints in order to create a better classifier than could be obtained from supervised methods applied to the labeled training set alone. Effective SSL imposes structural assumptions on the data, e.g. that neighbors are more likely to share a classification or that the decision boundary lies in an area of low density. For complex and high-dimensional data, neural networks can learn feature embeddings to which traditional SSL methods can then be applied in what we call hybrid methods. Previously-developed hybrid methods iterate between refining a latent representation and performing graph-based SSL on this representation. In this paper, we introduce a novel hybrid method that instead applies low-density separation to the embedded features. We describe it in detail and discuss why low-density separation may better suited for SSL on neural network-based embeddings than graph-based algorithms. We validate our method using in-house customer survey data and compare it to other state-of-the-art learning methods. Our approach effectively classifies thousands of unlabeled users from a relatively small number of hand-classified examples.

Keywords: Semi-supervised learning, low-density separation, deep learning, user classification from survey data

1 Background

In this section, we describe the problem of semi-supervised learning (SSL) from a mathematical perspective. We then outline some of the current approaches to solve this problem, emphasizing those relevant to our current work.

1.1 Problem Description

Consider a small labeled training set $\mathcal{D}_0 = \{(x_1, y_1), (x_2, y_2), \ldots, (x_\ell, y_\ell)\}$ of vector-valued features $x_i \in \mathbb{R}^d$ and discrete-valued labels $y_i \in \{1, \ldots, c\}$, for

---

⋆ Work completed while K.S. interned at Adobe.

$1 \leq i \leq \ell$. Suppose we have a large set $\mathcal{D}_1 = \{x_{\ell+1}, x_{\ell+2}, \ldots, x_{\ell+u}\}$ of unlabeled features to which we would like to assign labels. One could perform supervised learning on the labeled dataset $\mathcal{D}_0$ to obtain a general classifier and then apply this classifier to $\mathcal{D}_1$. However, this approach ignores any information about the distribution of the feature-points contained in $\mathcal{D}_1$. In contrast, SSL attempts to leverage this additional information in order to either inductively train a generalized classifier on the feature space or transductively assign labels only to the feature-points in $\mathcal{D}_1$.

Effective SSL methods impose additional assumptions about the structure of the feature-data (i.e., $\{x : (x, y) \in \mathcal{D}_0\} \cup \mathcal{D}_1$); for example, that features sharing the same label are clustered, that the decision boundary separating differently labeled features is smooth, or that the features lie on a lower dimensional manifold within $\mathbb{R}^d$. In practice, semi-supervised methods that leverage data from $\mathcal{D}_1$ can achieve much better performance than supervised methods that use $\mathcal{D}_0$ alone. See Figure 1 for a visualization. We describe both graph-based and low-density separation methods along with neural network-based approaches, as these are most closely related to our work. For a full survey, see [7,37,50].
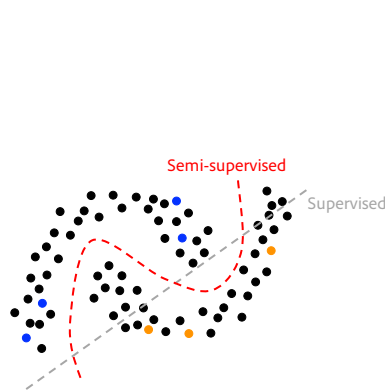

Figure 1. A schematic for semi-supervised classification. The grey line corresponds to a decision boundary obtained from a generic supervised classifier (incorporating information only from the labeled blue and orange points); the red line corresponds to a boundary from a generic semi-supervised method seeking a low-density decision boundary.
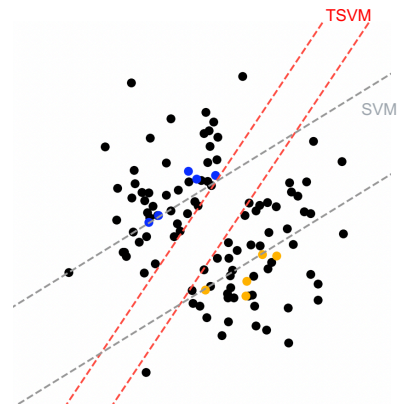
Figure 2. A schematic for TSVM segmentation. The grey lines correspond to maximum margin separation for labeled data using a standard SVM; the red lines correspond to additionally penalizing unlabeled points that lie in the margin. In this example, the data is perfectly separable in two dimensions, but this need not always be true.

1.2   Graph-Based Methods

Graph-based methods calculate the pairwise similarities between labeled and unlabeled feature-points and allow labeled feature-points to pass labels to their unlabeled neighbors. For example, label propagation [51] forms a $(\ell + u) \times (\ell + u)$ dimensional transition matrix T with transition probabilities proportional to similarities (kernelized distances) between feature-points and an $(\ell + u) \times c$ dimensional matrix of class probabilities, and (after potentially smoothing this matrix) iteratively sets $Y \leftarrow TY$, row-normalizes the probability vectors, and resets the rows of probability vectors corresponding to the already-labeled elements of $\mathcal{D}_0$. Label spreading [48] follows a similar approach but normalizes its weight matrix and allows for a (typically hand-tuned) clamping parameter that assigns a level of uncertainty to the labels in $\mathcal{D}_0$. There are many variations to the graph-based approach, including those that use graph min-cuts [4] and Markov random walks [40].

1.3   Low-Density Separation

Low-density separation methods attempt to find a decision boundary that best separates one class of labeled data from the other. The quintessential example is the transductive support vector machine (TSVM: [15,1,21,6,8,30]), a semi-supervised maximum-margin classifier of which there have been numerous variations. As compared to the standard SVM (*cf.*, e.g., [2,32]), the TSVM additionally penalizes unlabeled points that lie close to the decision boundary. In particular, for a binary classification problem with labels $y_i \in \{-1, 1\}$, it seeks parameters w, b that minimize the non-convex objective function

$$J(w,b) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l} H(y_i \cdot f_{w,b}(x_i)) + C^* \sum_{i=l+1}^{u} H(|f_{w,b}(x_i)|), \quad (1)$$

where $f_{w,b} : \mathbb{R}^d \to \mathbb{R}$ is the linear decision function $f_{w,b}(x) = w \cdot x + b$, and $H(x) = \max(0, 1 - x)$ is the hinge loss function. The hyperparameters C and C^* control the relative influence of the labeled and unlabeled data, respectively. Note that the third term, corresponding to a loss function for the unlabeled data, is non-convex, providing a challenge to optimization. See Figure 2 for a visualization of how the TSVM is intended to work and Ding et al. [13] for a survey of semi-supervised SVM's. Other methods for low-density separation include the more general entropy minimization approach [17], along with information regularization [39] and a Gaussian process-based approach [27].

1.4   Neural Network-Based Embeddings

Both the graph-based and low-density separation approaches to SSL rely on the geometry of the feature-space providing a reasonable approximation to the true underlying characteristics of the users or objects of interest. As datasets become increasingly complex and high-dimensional, Euclidean distance between

feature vectors may not prove to be the best proxy for user or item similarity. As the Gaussian kernel is a monotonic function of Euclidean distance, kernelized methods such as label propagation and label spreading also suffer from this criticism. While kernel learning approaches pose one potential solution [9,49], neural network-based embeddings have become increasingly popular in recent years. Variational autoencoders (VAE's: [24]) and generative adversarial nets (GAN's: [29,12]) have both been successfully used for SSL. However, optimizing the parameters for these types of networks can require expert hand-tuning and/or prohibitive computational expense [53,35]. Additionally, most research in the area concentrates on computer vision problems, and it is not clear how readily the architectures and techniques developed for image classification translate to other domains of interest.

1.5   Hybrid Methods

Recently, Iscen et al. introduced a neural embedding-based method to generate features on which to perform label propagation [19]. They train a neural network-based classifier on the supervised dataset and then embed all feature-points into an intermediate representation space. They then iterate between performing label propagation in this feature space and continuing to train their neural network classifier using weighted predictions from label propagation (see also [52]). As these procedures are similar in spirit to ours, we next outline our method in the next section and provide more details as part of a comparison in subsection 2.4.

2   Deep Low-Density Separation Algorithm

In this section, we provide a general overview of our algorithm for deep low-density separation and then delve into some of the details. We characterize our general process as follows:

1. We first learn a neural network embedding $f : \mathbb{R}^d \to \mathbb{R}^m$ for our feature-data optimized to differentiate between class labels. We define a network $g : \mathbb{R}^m \to \mathbb{P}^c$ (initialized as the initial layers from an autoencoder for the feature-data), where $\mathbb{P}^c$ is the space of c-dimensional probability vectors, and optimize $g \circ f$ on our labeled dataset $\mathcal{D}_0$, where we one-hot encode the categories corresponding to each y_i.
2. We map all of the feature-points through this deep embedding and then implement one-vs.-rest TSVM's for each class on this embedded data to learn class-propensities for each unlabeled data point. We augment our training data with the x_i from $\mathcal{D}_1$ paired with the propensities returned by this method and continue to train $g \circ f$ on this new dataset for a few epochs.
3. Our neural network f now provides an even better embedding for differentiating between classes. We repeat step 2 for a few iterations in order for the better embedding to improve TSVM separation, which upon further training yields an even better embedding, and so forth, etc.

Data: labeled dataset $\mathcal{D}_0$ and unlabeled dataset $\mathcal{D}_1$
Result: probabilistic predictions for the labels in $\mathcal{D}_1$
Initialize a deep neural network $f_\theta : \mathbb{R}^d \to \mathbb{R}^m$ with trainable parameters θ;
Initialize a neural network $g_\psi : \mathbb{R}^m \to \mathbb{P}^c$ with trainable parameters ψ;
Obtain θ_0, ψ_0 by minimizing cross entropy between $h(y_i)$ and $g_\psi(f_\theta(x_i))$ for
 $(x_i, y_i) \in \mathcal{D}_0$, where h is the encoding defined in (2);
for $t = 1, \ldots, T$ **do**
    Compute $\tilde{D}_0 = \{(f_{\theta_{t-1}}(x), y) : (x, y) \in \mathcal{D}_0\}$ and $\tilde{D}_1 = \{f_{\theta_{t-1}}(x) : x \in \mathcal{D}_1\}$;
    Perform one-vs.-rest TSVM training on $\tilde{\mathcal{D}}_0$ and $\tilde{\mathcal{D}}_1$ to obtain predicted
      probabilities $\hat{p}_i, i = \ell + 1, \ldots, \ell + u$ that the x_i in $\mathcal{D}_1$ lie in each class and
      then set $\breve{\mathcal{D}}_1 = \{(x_i, \hat{p}_i)\}$;
    Obtain θ_t, ψ_t by continuing to optimize $g_\psi \circ f_\theta$, using $\mathcal{D}_0 \cup \breve{\mathcal{D}}_1$;
end
return $g_{\psi_T}(f_{\theta_T}(x_i))$ or an exponential moving average of the probabilistic
 predictions $g_{\psi_t}(f_{\theta_t}(x_i))$ for $x_i \in \mathcal{D}_0$.

Algorithm 1: the Deep Segmentation Algorithm

This is our basic methodology, summarized as pseudo-code in Algorithm 1. Upon completion, it returns a neural network $g \circ f$ that maps feature-values to class/label propensities that can easily be applied to $\mathcal{D}_1$ and solve our problem of interest. In practice, we find that taking an exponentially decaying moving average of the returned probabilities as the algorithm progresses provides a slightly improved estimate. At each iteration of the algorithm, we reinitialize the labels for the unlabeled points and allow the semi-supervised TSVM to make inferences using the new embedding of the feature-data alone. In this way, it is possible to recover from mistakes in labeling that occurred in previous iterations of the algorithm.

2.1 Details: Neural Network Training

In our instantiation, the neural network $f : \mathbb{R}^d \to \mathbb{R}^m$ has two layers, the first of size 128 and the second of size 32, both with hyperbolic tangent activation. In between these two layers, we apply batch normalization [18] followed by dropout at a rate of 0.5 during model training to prevent overfitting [38]. The neural network $g : \mathbb{R}^m \to \mathbb{P}^c$ consists of a single layer with 5 units and softmax activation. We let θ (resp. ψ) denote the trainable parameters for f (resp. g) and sometimes use the notation f_θ and g_ψ to stress the dependence of the neural networks on these trainable parameters. Neural network parameters receive Glorot normal initialization [16]. The network weights for f and g receive Tikhonov-regularization [43,44], which decreases as one progresses through the network.

We form our underlying target distribution by one-hot encoding the labels y_i and slightly smoothing these labels. We define $h : \{1, \ldots, c\} \to \mathbb{P}^c$ by its

components $1 \le j \le c$ as

$$h(y)_j = \begin{cases} 1 - c \cdot \epsilon, & \text{if } y = j, \\ \epsilon, & \text{otherwise} \end{cases} \tag{2}$$

where we set $\epsilon = 10^{-3}$ to be our smoothing parameter.

Training proceeds as follows. We begin by training the neural network f_θ to minimize $D_{\mathrm{KL}}\big(h(y_i)||g_\psi(f_\theta(x_i))\big)$ the Kullback–Leibler (KL) divergence between the true distributions $h(y_i)$ and our inferred distributions $g_\psi(f_\theta(x_i))$, on $\mathcal{D}_0$ in batches. For parameter updates, we use the Adam optimizer [23] that maintains different learning rates for each parameter like AdaGrad [14] and allows these rates to sometimes increase like Adadelta [47] but adapts them based on the first two moments from recent gradient updates. This optimization on labeled data produces parameters θ_0 for f and ψ_0 for g.
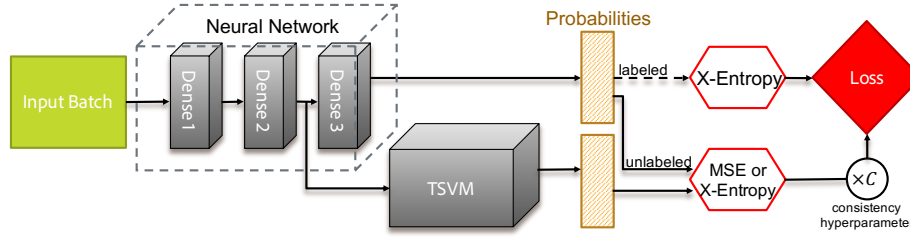

Figure 3. A schematic for Deep Low-Density Separation. The first two layers of the neural network correspond to f, the last to g. The semi-supervised model corresponds to the TSVM segmentation. We optimize on the unlabeled dataset using mean square error (MSE) and on the labeled dataset using cross-entropy (X-Entropy).

2.2   Details: Low-Density Separation

Upon initializing f and g, f_{θ_0} is a mapping that produces features well-suited to differentiating between classes. We form $\tilde{D}_0 = \{(f_{\theta_0}(x), y) : (x, y) \in \mathcal{D}_0\}$ and $\tilde{D}_1 = \{f_{\theta_0}(x) : x \in \mathcal{D}_1\}$ by passing the feature-data through this mapping. We then train c TSVM's, one for each class, on the labeled data $\tilde{D}_0$ and unlabeled data $\tilde{D}_1$.

Our implementation follows Collobert et al.'s TSVM-CCCP method [11] and is based on the R implementation in RSSL [25]. The algorithm decomposes the TSVM loss function $J(w, b)$ from (1) into the sum of a concave function and a convex function by creating two copies of the unlabeled data, one with positive labels and one with negative labels. Using the concave-convex procedure (CCCP: [46,45]), it then reduces the original optimization problem to an iterative procedure where each step requires solving a convex optimization problem similar to that of

the supervised SVM. These convex problems are then solved using quadratic programming on the dual formulations (for details, see [5]). Collobert et al. argue that TSVM-CCCP outperforms previous TSVM algorithms with respect to both speed and accuracy [11].

2.3   Details: Iterative Refinement

Upon training the TSVM's, we obtain a probability vector $\hat{p}_i \in \mathbb{P}^c$ for each $i = \ell+1, \ldots, \ell+u$ with elements corresponding to the likelihood that x_i lies in a given class. We then form $\check{\mathcal{D}}_1 = \{(x_i, \hat{p}_i)\}$ and obtain a supervised training set for further refining $g \circ f$. We set the learning rate for our Adam optimizer to 1/10th of its initial rate and minimize the mean square error between $g(f(x_i))$ and $\hat{p}_i$ for $(x_i, \hat{p}_i) \in \check{\mathcal{D}}_1$ for 10 epochs (*cf.* "consistency loss" from [42]) and then minimize the KL-divergence between $h(y_i)$ and $g(f(x_i))$ for 10 epochs. This training starts with neural network parameters θ_0 and ψ_0 and produces parameters θ_1 and ψ_1. Then, f_{θ_1} is a mapping that produces features better suited to segmenting classes than those from f_{θ_0}. We pass our feature-data through this mapping and continue the iterative process for $T = 6$ iterations. Our settings for learning rate, number of epochs, and T were hand-chosen for our data and would likely vary for different applications.

As the algorithm progresses, we store the predictions $g_{\psi_t}(f_{\theta_t}(x_i))$ at each step t and form an exponential moving average (discount rate $\rho = 0.8$) over them to produce our final estimate for the probabilities of interest.

2.4   Remarks on Methodology

We view our algorithm as most closely related to the work of Iscen et al. [19] and Zhuang et al. [52]. Both their work and ours iterate between refining a neural network-based latent representation and applying a classical SSL method to that representation to produce labels for further network training. While their work concentrates on graph-based label propagation, ours uses low-density separation, an approach that we believe may be more suitable for the task. The representational embedding we learn is optimized to discriminate between class labels, and for this reason we argue it makes more sense to refine decision boundaries than it does to pass labels. Additionally, previous work on neural network-based classification suggests that an SVM loss function can improve classification accuracy [41], and our data augmentation step effectively imposes such a loss function for further network training.

By re-learning decision boundaries at each iterative step, we allow our algorithm to recover from mistakes it makes in early iterations. One failure mode of semi-supervised methods entails making a few false label assignments early in the iterative process and then having these mislabeled points pass these incorrect labels to their neighbors. For example, in pseudo-labelling [28], the algorithm augments the underlying training set $\mathcal{D}_0$ with pairs $(x_i, \hat{y}_i)$ for $x_i \in \mathcal{D}_1$ and predicted labels $\hat{y}_i$ for which the model was most confident in the previous iteration. Similar error-reinforcement problems can occur with boosting [31]. It is easy to

see how a few confident, but inaccurate, labels that occur in the first few steps of the algorithm can set the labeling process completely askew.

By creating an embedding $f : \mathbb{R}^d \to \mathbb{R}^m$ and applying linear separation to embedded points, we have effectively learned a distance metric $\kappa : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_{\geq 0}$ especially suited to our learning problem. The linear decision boundaries we produce in $\mathbb{R}^m$ correspond to nonlinear boundaries for our original features in $\mathbb{R}^d$. Previously, Jean et al. [20] described using a deep neural network to embed features for Gaussian process regression, though they use a probabilistic framework for SSL and consider a completely different objective function.

3   Application to User Classification from Survey Data

In this section, we discuss the practical problem of segmenting users from survey data and compare the performance of our algorithm to other recently-developed methods for SSL on real data. We also perform an ablation study to ensure each component of our process contributes to the overall effectiveness of the algorithm.

3.1   Description of the Dataset

At Adobe, we are interested in segmenting users based on their work habits, artistic motivations, and relationship with creative software. To gather data, we administered detailed surveys to a select group of users in the US, UK, Germany, & Japan (just over 22 thousand of our millions of users). We applied Latent Dirichlet Allocation (LDA: [33,3]), an unsupervised model to discover latent topics, to one-hot encoded features generated from this survey data to classify each surveyed user as belonging to one of $c = 5$ different segments. We generated profile and usage features using an in-house feature generation pipeline (that could in the future readily be used to generate features for the whole population of users). In order to be able to evaluate model performance, we masked the LDA labels from our surveyed users at random to form the labelled and unlabelled training sets $\mathcal{D}_0$ and $\mathcal{D}_1$.

3.2   State-of-the-Art Alternatives

We compare our algorithm against two popular classification algorithms. We focus our efforts on other algorithms we might have actually used in practice instead of more similar methods that just recently appeared in the literature.

The first, LightGBM [22] is a supervised method that attempts to improve upon other boosted random forest algorithms (e.g. the popular xgBoost [10]) using novel approaches to sampling and feature bundling. It is our team's preferred nonlinear classifier, due to its low requirements for hyperparameter tuning and effectiveness on a wide variety of data types. As part of the experiment, we wanted to evaluate the conditions for semi-supervised learning to outperform supervised learning.

The second, Mean Teacher [42] is a semi-supervised method that creates two supervised neural networks, a teacher network and a student network, and trains both networks using randomly perturbed data. Training enforces a consistency loss between the outputs (predicted probabilities in $\mathbb{P}^c$) of the two networks: optimization updates parameters for the student network and an exponential moving averages of these parameters become the parameters for the teacher network. The method builds upon Temporal Ensembling [26] and uses consistency loss [34,36].

3.3   Experimental Setup

We test our method with labelled training sets of successively increasing size $\ell \in \{35, 50, 125, 250, 500, 1250, 2500\}$. Each training set is a strict superset of the smaller training sets, so with each larger set, we strictly increase the amount of information available to the classifiers. To tune hyperparameters, we use a validation set of size 100, and for testing we use a test set of size 4780. The training, validation, and test sets are selected to all have equal class sizes.

For our algorithm, we perform $T = 6$ iterations of refinement, and in the TSVM we set the cost parameters $C = 0.1$ and $C^* = \frac{\ell}{u}C$. To reduce training time, we subsample the unlabeled data in the test set by choosing 250 unlabeled points uniformly at random to include in the TSVM training. We test using our own implementations of TSVM and MeanTeacher.

3.4   Numerical Results and Ablation

Table 1 reports our classification accuracy on five randomized shuffles of the training, validation, and test sets. These results are summarized in Figure 4. The accuracy of our baseline methods are shown first, followed by three components of our model:

1. Initial NN: The output of the neural network after initial supervised training.
2. DeepSep-NN: The output of the neural network after iterative refinement with Algorithm 1.
3. DeepSep-Ensemble: Exponential moving average as described in Algorithm 1.

We find that Deep Low-Density Separation outperforms or matches LightGBM in the range $\ell \leq 1250$. The relative increase in accuracy of Deep Separation is as much as 27%, which is most pronounced with a very small amount of training data ($\ell \leq 50$). Some of this increase can be attributed to the initial accuracy of the neural network; however, the iterative refinement of Deep Separation improves the accuracy of the initial network by up to 8.3% (relative). The addition of a temporal ensemble decreases variance in the final model, further increasing accuracy by an average of 0.54% across the range. Compared to Mean Teacher, the iterative refinement of Deep Separation achieves a larger increase in accuracy for $l \leq 500$.

To visualize how the iterative refinement process and exponential weighted average improve the model, Figure 5 shows the accuracy of our model at each
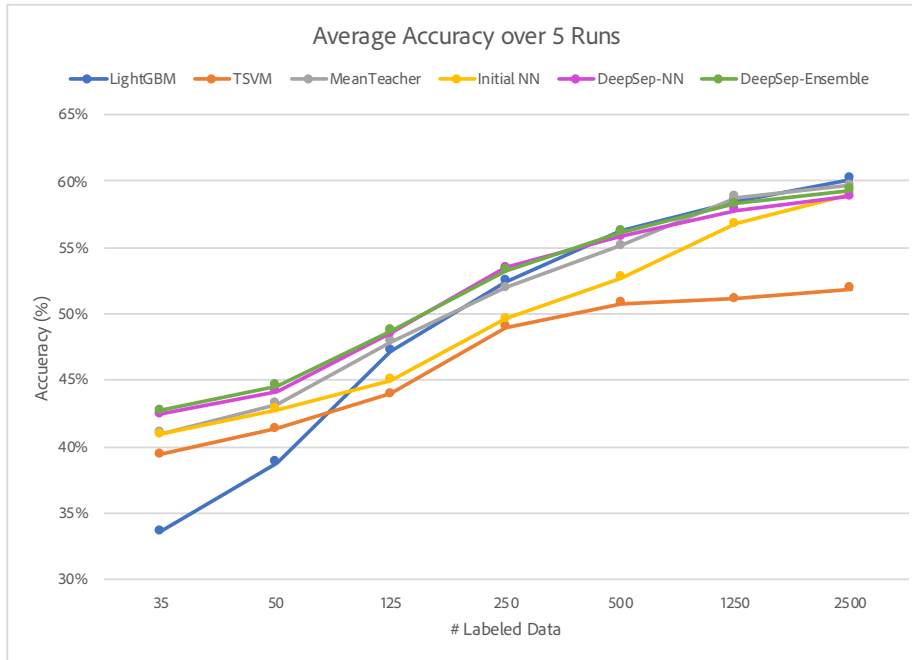
Figure 4. Average accuracy over 5 random shuffles for LightGBM, TSVM, MeanTeacher and our proposed method. Random chance accuracy is 20%. We are primarily interested in the regime where few training examples exist – particularly when the number of labeled datapoints is 35-50.

iteration. We see that for each random shuffle, the refinement process leads to increased accuracy compared to the initial model. However, the accuracy of the neural network fluctuates by a few percent at a time. Applying the exponential moving average greatly reduces the impact of these fluctuations and yields more consistent improvement, with a net increase in accuracy on average.

Regarding training time, all numerical experiments were performed on a mid-2018 MacBook Pro (2.6 GHz Intel Core i7 Processor; 16 GB 2400 MHz DDR4 Memory). Deep Separation takes up to half an hour on the largest training set ($\ell = 2500$). However, we note that for $\ell \leq 500$, the model takes at most three minutes, and this is the regime where our method performs best in comparison to other methods. In contrast, LightGBM takes under a minute to run with all training set sizes.

4   Conclusions

In this paper, we introduce a novel hybrid semi-supervised learning method, Deep Low-Density Separation, that iteratively refines a latent feature representation and
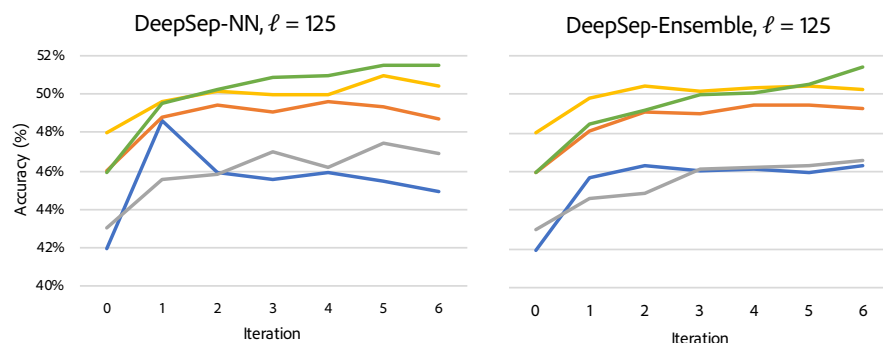
Figure 5. Classification accuracy on the test set for all five random shuffles over the course of iterative refinement, using 125 labeled data, of (left) the refined neural network and (right) the exponential moving average of predictions. Here, different colors correspond to different choices for training set (different random seeds).

then applies low-density separation to this representation to augment the training set. We validate our method on a multi-segment classification dataset generated from surveying Adobe's user base. In the future, we hope to further investigate the interplay between learned feature embeddings and low-density separation methods, and experiment with different approaches for both representational learning and low-density separation. While much of the recent work in deep SSL concerns computer vision problems and image classification in particular, we believe these methods will find wider applicability within academia and industry, and anticipate future advances in the subject.

References

1. Bennett, K.P., Demiri, A.: Semi-supervised support vector machines. In: Adv. Neural Inf. Process. Syst. (1998)
2. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer (2006)
3. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet allocation. J. Mach. Learn. Res. **3**, 993–1022 (2003)
4. Blum, A., Chawla, S.: Learning from labeled and unlabeled data using graph mincuts. In: Int. Conf. Mach. Learn. pp. 19–26 (2001)
5. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge U. Press (2004)
6. Chapelle, O., Zien, A.: Semi-supervised classification by low density separation. In: Conf. Artif. Intell. Stats. (2005)
7. Chapelle, O., Schölkopf, B., Zien, A. (eds.): Semi-Supervised Learning. MIT Press (2006)
8. Chapelle, O., Sindhwani, V., Keerthi, S.S.: Optimization techniques for semi-supervised support vector machines. J. Mach. Learn. Res. pp. 203–233 (2008)
9. Chapelle, O., Weston, J., Schölkopf, B.: Cluster kernels for semi-supervised learning. In: Adv. Neural Inf. Process. Syst. pp. 601–608 (2003)

10. Chen, T., Guestrin, C.: XGBoost: A scalable tree boosting system. In: Int. Conf. Knowl. Discov. Data Min. pp. 785–794 (2016)
11. Collobert, R., Sinz, F., Weston, J., Bottou, L.: Large scale transductive SVMs. J. Mach. Learn. Res. **7**, 1687–1712 (2006)
12. Dai, Z., Yang, Z., Yang, F., Cohen, W.W., Salakhutdinov, R.: Good semi-supervised learning that requires a bad GAN. In: Adv. Neural Inf. Process. Syst. pp. 6513–6523 (2017)
13. Ding, S., Zhu, Z., Zhang, X.: An overview on semi-supervised support vector machine. Neural Comput. & Applic. **28**(5), 969–978 (2017)
14. Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. J. Mach. Learn. Res. **12**, 2121–2159 (2011)
15. Gammerman, A., Vovk, V., Vapnik, V.: Learning by transduction. In: Uncertain. Artif. Intell. pp. 148–155 (1998)
16. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Conf. Artif. Intell. Stats. vol. 9, pp. 249–256 (2010)
17. Grandvalet, Y., Bengio, Y.: Semi-supervised learning by entropy minimization. In: Adv. Neural Inf. Process. Syst. pp. 529–536 (2004)
18. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Int. Conf. Mach. Learn. pp. 448–456 (2015)
19. Iscen, A., Tolias, G., Avrithis, Y., Chum, O.: Label propagation for deep semi-supervised learning. In: Conf. Comput. Vis. Pattern Recognit. (2019)
20. Jean, N., Xie, S.M., Ermon, S.: Semi-supervised deep kernel learning: Regression with unlabeled data by minimizing predictive variance. In: Adv. Neural Inf. Process. Syst. pp. 5322–5333 (2018)
21. Joachims, T.: Transductive inference for text classification using support vector machines. In: Int. Conf. Mach. Learn. pp. 200–209 (1999)
22. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T.Y.: LightGBM: A highly efficient gradient boosting decision tree. In: Adv. Neural Inf. Process. Syst. pp. 3146–3154 (2017)
23. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. Int. Conf. Learn. Represent. (2015)
24. Kingma, D.P., Mohamed, S., Rezende, D.J., Welling, M.: Semi-supervised learning with deep generative models. In: Adv. Neural Inf. Process. Syst. pp. 3581–3589 (2014)
25. Krijthe, J.H.: RSSL: Semi-supervised learning in R. In: Reprod. Res. Pattern Recognit. pp. 104–115 (2017)
26. Laine, S., Aila, T.: Temporal ensembling for semi-supervised learning. In: Int. Conf. Learn. Represent. (2017)
27. Lawrence, N.D., Jordan, M.I.: Semi-supervised learning via gaussian processes. In: Adv. Neural Inf. Process. Syst. pp. 753–760 (2005)
28. Lee, D.H.: Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In: ICML Workshop on Challenges in Representation Learning (2013)
29. Li, C., Xu, T., Zhu, J., Zhang, B.: Triple generative adversarial nets. In: Adv. Neural Inf. Process. Syst. pp. 4088–4098 (2017)
30. Li, Y., Zhou, Z.: Towards making unlabeled data never hurt. IEEE Trans. Pattern Anal. Mach. Intell. **37**(1), 175–188 (2015)
31. Mallapragada, P.K., Jin, R., Jain, A.K., Liu, Y.: Semiboost: Boosting for semi-supervised learning. IEEE Trans. Pattern Anal. Mach. Intell. **31**(11), 2000–2014 (2009)

32. Murphy, K.P.: Machine Learning: A Probabilistic Perspective. MIT Press (2012)
33. Pritchard, J.K., Stephens, M., Donnelly, P.: Inference of population structure using multilocus genotype data. Genetics **155**(2), 945–959 (2000)
34. Rasmus, A., Berglund, M., Honkala, M., Valpola, H., Raiko, T.: Semi-supervised learning with ladder networks. In: Adv. Neural Inf. Process. Syst. pp. 3546–3554 (2015)
35. Real, E., Moore, S., Selle, A., Saxena, S., Suematsu, Y.L., Le, Q., Kurakin, A.: Large-scale evolution of image classifiers. Int. Conf. Mach. Learn. (2017)
36. Sajjadi, M., Javanmardi, M., Tasdizen, T.: Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In: Adv. Neural Inf. Process. Syst. pp. 1163–1171 (2016)
37. Seeger, M.: Learning with labeled and unlabeled data. Tech. rep., U. Edinburgh (2001)
38. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**, 1929–1958 (2014)
39. Szummer, M., Jaakkola, T.: Information regularization with partially labeled data. In: Adv. Neural Inf. Process. Syst. pp. 1049–1056 (2002)
40. Szummer, M., Jaakkola, T.: Partially labeled classification with markov random walks. In: Adv. Neural Inf. Process. Syst. pp. 945–952 (2002)
41. Tang, Y.: Deep learning using linear support vector machines. In: Int. Conf. Mach. Learn.: Challenges in Representation Learning Workshop (2013)
42. Tarvainen, A., Valpola, H.: Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In: Adv. Neural Inf. Process. Syst. pp. 1195–1204 (2017)
43. Tikhonov, A.N.: On the stability of inverse problems. Proc. USSR Acad. Sci. **39**(5), 195–198 (1943)
44. Tikhonov, A.N.: Solution of incorrectly formulated problems and the regularization method. Proc. USSR Acad. Sci. **151**(3), 501–504 (1963)
45. Yuille, A.L., Rangarajan, A.: The concave-convex procedure. Neural Comput. **15**(4), 915–936 (2003)
46. Yuille, A.L., Rangarajan, A.: The concave-convex procedure (CCCP). In: Adv. Neural Inf. Process. Syst. pp. 1033–1040 (2002)
47. Zeiler, M.D.: Adadelta: An adaptive learning rate method (2012), arXiv:1212.5701
48. Zhou, D., Bousquet, O., Lal, T.N., Weston, J., Schölkopf, B.: Learning with local and global consistency. In: Adv. Neural Inf. Process. Syst. pp. 321–328 (2004)
49. Zhu, J., Kandola, J., Ghahramani, Z., Lafferty, J.D.: Nonparametric transforms of graph kernels for semi-supervised learning. In: Adv. Neural Inf. Process. Syst. pp. 1641–1648 (2005)
50. Zhu, X.: Semi-supervised learning literature survey. Tech. rep., U. Wisconsin-Madison (2005)
51. Zhu, X., Ghahramani, Z.: Learning from labeled and unlabeled data with label propagation. Tech. Rep. CMU-CALD-02-107, Carnegie Mellon U. (2002)
52. Zhuang, C., Ding, X., Murli, D., Yamins, D.: Local label propagation for large-scale semi-supervised learning (2019), arXiv:1905.11581
53. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. Int. Conf. Mach. Learn. (2017)

Table 1. Classification accuracy (in percent) for each of the methods tested. Shuffle # refers to the randomized splitting of the data into training, validation, and test sets. The final block contains the average accuracy over 5 random shuffles.

Shuffle #	Model	ℓ						
		35	50	125	250	500	1250	2500
1	LightGBM	30.98	34.73	47.45	51.55	55.99	59.39	60.65
	TSVM	38.65	38.26	40.26	46.84	48.54	51.02	52.94
	MeanTeacher	39.91	41.70	47.54	51.33	54.81	59.83	60.48
	Initial NN	38.65	40.09	41.92	47.89	51.15	58.13	61.09
	DeepSep-NN	39.04	41.79	44.97	53.55	54.51	57.60	60.13
	DeepSep-Ensemble	40.13	42.00	46.32	52.68	54.95	58.04	59.87
2	LightGBM	32.03	38.30	46.93	52.33	55.86	58.39	59.61
	TSVM	43.31	43.88	47.45	49.19	50.76	49.76	50.37
	MeanTeacher	43.14	42.75	48.58	53.03	54.12	58.08	59.35
	Initial NN	43.31	43.79	45.97	50.72	53.03	57.04	59.26
	DeepSep-NN	47.32	47.10	48.85	51.90	54.25	56.69	57.95
	DeepSep-Ensemble	46.45	46.58	49.06	51.94	54.47	57.60	58.56
3	LightGBM	32.33	40.31	47.63	50.94	56.34	57.82	60.13
	TSVM	30.37	34.55	37.30	49.93	51.59	52.42	51.42
	MeanTeacher	35.77	40.26	45.05	50.33	55.12	56.43	57.25
	Initial NN	37.12	40.87	43.05	48.15	52.72	55.82	57.86
	DeepSep-NN	36.69	40.48	46.88	52.33	55.60	56.95	57.82
	DeepSep-Ensemble	37.17	40.52	46.49	52.33	56.12	57.04	57.86
4	LightGBM	35.12	36.17	47.36	52.42	56.30	59.00	61.05
	TSVM	40.61	45.10	48.28	52.85	52.64	50.11	51.29
	MeanTeacher	41.96	44.31	49.54	51.76	55.56	59.56	60.96
	Initial NN	41.26	43.66	48.10	48.63	52.55	55.64	58.26
	DeepSep-NN	44.84	44.58	50.41	54.34	56.86	58.61	59.08
	DeepSep-Ensemble	44.49	44.88	50.33	53.46	56.86	59.39	60.44
5	LightGBM	37.60	44.44	46.67	55.16	56.60	57.95	59.30
	TSVM	44.14	45.14	46.71	46.06	50.41	52.24	53.51
	MeanTeacher	44.14	46.93	48.63	53.25	56.08	60.17	60.22
	Initial NN	44.44	45.62	45.88	52.85	54.29	57.39	58.69
	DeepSep-NN	44.44	46.93	51.46	55.38	58.00	59.39	59.17
	DeepSep-Ensemble	45.53	48.85	51.37	55.90	58.43	59.48	59.96
Average	LightGBM	33.61	38.79	47.21	52.48	56.22	58.51	60.15
	TSVM	39.42	41.39	44.00	48.98	50.79	51.11	51.90
	MeanTeacher	40.98	43.19	47.87	51.94	55.14	58.81	59.65
	Initial NN	40.96	42.81	44.98	49.65	52.75	56.80	59.03
	DeepSep-NN	42.47	44.17	48.51	53.50	55.84	57.85	58.83
	DeepSep-Ensemble	42.75	44.57	48.71	53.26	56.17	58.31	59.34