Vol.2012-NL-207 No.13
2012/7/27

Feature Scaling for Online Learning of Binary Classifiers

Danushka Bollegala[1,a)] Hitoshi Iba[1,b)]

Abstract: Scaling feature values is an important task in numerous machine learning tasks. Often feature scaling is conducted in an unsupervised manner as a preprocessing task prior to learning. By conducting feature scaling at train time, we can quickly adapt to the changes in data stream as well as scale features to optimize classification accuracy. We study the effect of feature scaling at training time for one-pass online binary classification algorithms. We propose a joint supervised feature scaling method to simultaneously scale features during training time. We incorporate the proposed method into a binary logistic regression model and train using stochastic gradient descent in a one-pass online learning setting.

Keywords: Feature scaling, Online learning, Binary classification.

1. Introduction

Machine learning algorithms require train and test instances to be represented using a set of features. For example, in supervised document classification [6], a document is often represented as a vector of its words and the value of a feature is set to the number of times the word corresponding to the feature occurs in that document. However, different features occupy different value ranges, and often one must scale the feature values before any supervised classifier is trained. In our example of document classification, there are both highly frequent words (e.g. stop words) as well as extremely rare words. Often, the relative difference of a value of a feature is more informative than its absolute value. Therefore, feature scaling has shown to improve performance in classification algorithms.

Typically, feature values are scaled to a fixed range in a preprocessing step before using the scaled features in the subsequent learning task. However, this preprocessing approach to feature value scaling is problematic because of several reasons. First, often feature scaling is done in an unsupervised manner without consulting the labels assigned to the training instances. Although this is the only option in unsupervised learning tasks such as document clustering, for supervised learning tasks such as document classification, where we do have access to the label information, we can use the label information also for feature scaling. Second, it is not possible to perform feature scaling as a preprocessing step in *one-pass* online learning setting. In one-pass online learning we are allowed to traverse through the set of training instances only once. Learning from extremely large datasets such as twitter streams or Web scale learning calls for algorithms that require only a single pass over the set of training instances. In such scenarios it is not possible to scale the feature values beforehand by using statistics regarding the entire training set and then perform supervised learning using the scaled feature vectors. Simple feature scaling methods such as fitting a zero mean and unit variance Gaussian for each feature requires a large sample of training instances to accurately compute the mean and the standard deviation for each feature, and cannot be performed only using a single training instance as in the case of one-pass online learning. Third, even if we pre-compute scaling parameters for a feature, those values might become irrelevant in an online learning setting in which properties of the training instances vary over the time. For example, a twitter text stream regarding a particular keyword might change overtime and the scaling factors computed using old data might not be appropriate for the new data.

We propose a joint supervised feature scaling method that can learn both the optimal feature scalings as well as the weight of each feature in an online binary classification setting. The proposed algorithm can be trained under the one-pass online learning setting, where only a single training instance is provided at a time and only the scale parameters and feature weights are stored in the memory. In particular, we do not require any preprocessing of the training instances prior to training. This enables the proposed method to (a) efficiently adapt to the varying statistics in the data stream, (b) compute the optimal feature scales such that the likelihood of the training data under the trained model is maximized, and (c) train from large datasets where batch learning is impossible because of memory requirements. We present our learning algorithm for the case of binary classification. Extending it to multi-class and regression tasks is our future work. We perform an extensive evaluation using numerous popular online learning algorithm and several variants of the proposed method, including an unsupervised online feature scaling method. It turns out that the unsupervised method outperforms supervised methods for feature scaling.

[1] Graduate School of Information Science and Technology, The University of Tokyo, 7-3-1, Hongo, Bunkyo-ku, Tokyo, 113–8656, Japan
[†1] Presently with The University of Tokyo
[a)] danushka@iba.t.u-tokyo.ac.jp
[b)] iba@iba.t.u-tokyo.ac.jp

2. One-Pass Online Learning

Before we proceed to explain the proposed feature scaling method, we must first state the problem setting, *One-Pass Online Learning*. In the One-Pass Online Learning (**OPOL**) we impose the restriction that *only a single-pass is allowed over the set of train instances* by the learning algorithm. Moreover, because we are interested in the online learning algorithms, we assume that we can process only a single feature vector at train or test time. The OPOL setting is more restrictive than the classical online learning setting where a learning algorithm is allowed to traverse multiple times over the training dataset. However, OPOL becomes the only possible approach in following scenarios.

(1) The number of instances in the training dataset is so large that it is impossible to traverse multiple times over the dataset.

(2) The dataset is in fact a stream where we encounter new instances continuously. For example, consider the situation where we want to train a sentiment classifier from tweets.

(3) The data stream changes over time. In this case, even if we can store old data instances they might not be much of a help to predict the latest trends in the data stream.

It must be noted that OPOL is not the only solution for the first scenario where we have a large training dataset. One alternative approach is to select a subset of example from the dataset at each iteration and only use that subset for training in that iteration. One promising criterion for selecting examples for training is curriculum learning [1]. In curriculum learning, a learner is presented with easy examples first and gradually with the more difficult examples. However, determining the criteria for selecting easy examples is a difficult problem itself, and the criterion for selecting easy examples might be different from one task to another. Moreover, it is not clear whether we can select easy examples from the training dataset in a sequential manner as required by online learning without consulting the unseen training examples.

The requirement for OPOL ever increases with the large training datasets and data streams we encounter on the Web such as social feeds. Most online learning algorithms require several passes over the training dataset to achieve convergence. For example, Passive-Aggressive algorithms require at least 5 iterations where as, for Confidence-Weighted algorithms the number of iterations has shown to be less (ca. 2). Our focus in this paper is not to develop online learning algorithms that can classify instances with high accuracy by traversing only once over the dataset, but to study the effect of feature scaling in the OPOL setting. To this end, we study both an unsupervised feature scaling method (Section 3) and several variants of a joint supervised feature scaling methods (Section 4).

3. Unsupervised Feature Scaling for OPOL

Before we discuss supervised feature scaling methods in Section 4, it is worthwhile to introduce a simple yet effective unsupervised approach to feature scaling. In this unsupervised approach, given a feature x_j, we compute the mean, $E(x_j)$ and the standard deviation $\sqrt{(V(x_j)}$ of the feature and perform an affine transformation as follows,

$$x'_j = \frac{x_j - E(x_j)}{\sqrt{V(x_j)}}. \tag{1}$$

This scaling operation corresponds to a linear shift of the feature values by the mean value of the feature, followed up by a scaling by its standard deviation. From a geometric point of view, this transformation will shift the origin to the mean value and then scale axis corresponding to the j-th feature to unit standard deviation. It is used popularly in batch learning setting, in which one can compute the mean and the standard deviation using *all* the training instances in the training dataset. However, this is not possible in OPOL, in which we encounter only one instance at a time. However, even in the OPOL setting, we can compute the mean and the standard deviation on the fly and constantly update our estimates of those values as new training instances (feature vectors) are observed. The update equations for the mean M_j^k and the standard deviation $\sqrt{S_j^k/(k-1)}$ for the j-th feature are as follows [4], [12],

$$\begin{aligned} M_j^k &= M_j^{k-1} + \frac{x_j^k - M_j^{k-1}}{k}, & (2)\\ S^k &= S^{k-1} + (x_j^k - M_j^{k-1})(x_j^k - M_j^k). & (3)\end{aligned}$$

As we will later see in Section 6, this simple unsupervised approach to feature scaling significantly outperforms all the supervised feature scaling approaches described in Section 4.

4. Supervised Feature Scaling for OPOL

We define the supervised feature scaling task for binary classification in the one-pass online learning setting as follows. Given a stream of labeled training instances $(\boldsymbol{x}_n, t_n)$, in which the class label t_n of the n-th training instance x_n, denoted by a feature vector $\boldsymbol{x}_n$, is assumed to be either $+1$ (positive class) or -1 (negative class). Furthermore, let us assume that the feature space is M dimensional and the value of the i-th feature of the n-th instance in the training data stream is denoted by x_i^n. In this paper, we consider only real-valued features (i.e. $x_i^n \in \mathbb{R}$) because feature scaling is particularly important for real-valued features.

We define the feature scaling function $\sigma_i(x_i^n)$ for the i-th feature as a function that maps $\mathbb{R}$ to the range $[0, 1]$ as follows:

$$\sigma_i(x_i^n) = \frac{1}{1 + \exp(-\alpha_i x_i^n + \beta_i)}. \tag{4}$$

Here, α_i and β_i are the scaling parameters for the i-th dimension of the feature space. Several important properties of the feature scaling function defined by Equation 4 must be noted. First, the feature transformation function maps all feature values to the range $[0, 1]$ irrespective of the original range in which each feature value x_i was. For example, one feature might originally be limited to the range $[0, 0.001]$, whereas another feature might have values in the full range of $[0, 10000]$. By scaling each feature into a common range we can concentrate on the relative values of those features without being biased by their absolute values. Second, the scaling parameters α_i and β_i are defined per-feature basis. This enables us to scale different features using scale parameters appropriate for their value ranges. Third, the

the linear transformation $\alpha_i x_i^n - \beta_i$ within the exponential term of the feature scaling function resembles the typical affine transformations performed in unsupervised feature scaling. For example, assuming the mean and the standard deviation of the i-th feature to be respectively μ_i and δ_i, in supervised classification, features are frequently scaled to $(x_i - \mu_i)/\delta_i$ prior to training and testing. The linear transformation within the exponential term in Equation 4 can be seen as a special case of this approach with values $\alpha_i = 1/\delta_i$ and $\beta_i = \mu_i/\delta_i$.

Then, the posterior probability, $P(t = 1|\boldsymbol{x}^n, b, \boldsymbol{\alpha}, \boldsymbol{\beta})$ of $\boldsymbol{x}^n$ belonging to the positive class is given as follows according to the logistic regression model [3]:

$$P(t_n = 1|\boldsymbol{x}^n, b, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{1 + \exp\left(-\sum_{i=1}^{M} w_i \sigma_i(x_i^n) - b\right)}, \quad (5)$$

$$P(t_n = 1|\boldsymbol{x}^n, b, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{1 + \exp\left(-\frac{w_i}{1+\exp(-\alpha_i x_i^n + \beta_i)} - b\right)}.$$

Here, w_i is the weight associated with the i-th feature and $b \in \mathbb{R}$ is the bias term. We arrange the weights w_i, scaling parameters α_i and β_i respectively using $\mathbb{R}^M$ vectors $\boldsymbol{w}$, $\boldsymbol{\alpha}$, and $\boldsymbol{\beta}$.

The cross-entropy loss function per instance including the $L2$ regularization terms for the weight vector $\boldsymbol{w}$ and scale vector $\boldsymbol{\beta}$ can be written as follows:

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}, \boldsymbol{\beta}) = -t_n \log y_n - (1 - t_n) \log(1 - y_n) \quad (6)$$

Here, we used $y_n = P(t = 1|\boldsymbol{x}^n, b, \boldsymbol{\alpha}, \boldsymbol{\beta})$ to minimize the cluttering of symbols in the Equation. To avoid overfitting to training instances and to minimize the distortion of the training instances we impose L2 regularization on $\boldsymbol{w}$, $\boldsymbol{\alpha}$, and $\boldsymbol{\beta}$. Therefore, the final objective function that must be minimized with respect to $\boldsymbol{w}$, $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and b is give by,

$$E(\boldsymbol{w}, b, \boldsymbol{\alpha}, \boldsymbol{\beta}) = L(\boldsymbol{w}, b, \boldsymbol{\alpha}, \boldsymbol{\beta}) + \lambda \|w\|_2 + \mu \|\alpha\|_2 + \nu \|\beta\|_2 \quad (7)$$

Here, λ, μ and ν respectively are the L2 regularization coefficients corresponding to the weight vector $\boldsymbol{w}$ and the scale vectors $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$. Because we consider the minimization of Equation 7 per instance basis, in our experiments we divide the regularization parameters λ, μ, and ν by the total number of training instances N in the dataset such that we can compare the values those parameters across datasets of different sizes.

By setting the partial derivatives $\frac{\partial E}{\partial w_j}$, $\frac{\partial E}{\partial b}$, $\frac{\partial E}{\partial \alpha_j}$, and $\frac{\partial E}{\partial \beta_j}$ to zero and applying Stochastic Gradient Descent (SGD) update rule the following updates can be derived,

$$w_j^{k+1} = w_j^k(1 - 2\lambda\eta_k) + \eta_k(t_n - y_n)\sigma_j(x_j^n), \quad (8)$$

$$b^{k+1} = b^k + \eta_k(t_n - y_n), \quad (9)$$

$$\alpha_j^{k+1} = \alpha_j^k(1 - 2\mu\eta_k) + \eta_k x_j^n w_j \sigma_j(x_j^n)(1 - \sigma_j(x_j^n))(t_n - y_n)(10)$$

$$\beta_j^{k+1} = \beta_j^k(1 - 2\nu\eta_k) - \eta_k(t_n - y_n) w_j \sigma_j(x_j^n)(1 - \sigma_j(x_j^n)). \quad (11)$$

In Equations 8-11, k denotes the k-th update and η_k is the learning rate for the k-th update. We experimented with both linear and exponential decaying and found linear decaying to perform better for the proposed method. The linear decaying function for η_k is defined as follows,

$$\eta_k = \frac{\eta_0}{1 + \frac{k}{T \times N}}. \quad (12)$$

Here, T is the total number of iterations for which the training dataset containing N instances will be traversed. Because we are considering OPOL, $T = 1$. The initial learning rate η_0 is set to 0.1 throughout the experiments described in the paper. This value of 0.1 was found to be producing the best results in our preliminary experiments using development data, which we selected randomly from the benchmark datasets described later in Section 5.

Several observations are in order. First, note that the scaling factors α_j and β_j distort the original value of the feature x_i. If this distortion is too much, then we might loose the information conveyed by the feature x_i. To minimize the distortion of $\boldsymbol{x}$ because of scaling, we have imposed regularization on both $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. This treatment is similar to the slack variables often used in non-separable classification tasks and imposing a penalty on the total slackness. Of course, the regularization on $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ can be removed simply by setting the corresponding regularization coefficients μ and ν to zero. Therefore, the introduction of regularization on $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ does not harm the generality of the proposed method. The total number of parameters to train in this model is $M + M + M + 1 = 3M + 1$, corresponding to $\boldsymbol{w}$, $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and b. Note that we must not regularize the bias term b and let it to adjust arbitrarily large. Doing so also act as a dynamic scaling for the score (i.e. inner-product between $\boldsymbol{w}$ and $\boldsymbol{x}$), although this type of scaling is *not* feature specific.

The feature scaling function given by Equation 4 is by no means the only function that satisfies the requirement for a scaling function (i.e. maps all feature values to the same range such as [0, 1]). Next, we introduce several important variants of Equation 4 and present the update equations for each variant. We omit the derivation of the update equations due to the limited availability of space. In Section 6, we empirically study the effect of the different variants discussed in the paper. For the ease of reference, we name the original formulation given by Equation 4 as **FS** (Feature Scaling) method.

The objective function given by Equation 7 is convex with respect to $\boldsymbol{w}$. This can be easily verified by computing the second derivative of the objective function with respect to w_i, which becomes

$$\frac{\partial^2 E}{\partial w_i^2} = \sigma(x_i)^2 y_n(1 - y_n) + 2\lambda. \quad (13)$$

Because $0 < \sigma(x_i) < 1$, $0 < y_n < 1$, and $0 < \lambda$ hold, the second derivative $\frac{\partial^2 E}{\partial w_i^2} > 0$, which proves that the objective function is convex with respect to w_i. Likewise, the objective function can be shown to be convex with respect to the bias term b. It is interesting to note that the convexity holds irrespective of the form of the scaling function σ for both $\boldsymbol{w}$ and b as long as $\sigma(x_i) \neq 0$ satisfies If $\sigma(x_i) = 0$ for some value of x_i, then the convexity of E also depends upon λ not being equal to zero. Although, in the case of sigmoid feature scaling functions $\sigma(x_i) \to 0$ when $x_i \to -\infty$ this is irrelevant because feature values are finite in practice.

Unfortunately, the objective function is *non-convex* with respect to $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. Although SGD updates are empirically shown to work well even when the objective function is non-convex, there is no guarantee that the update Equations 8 - 11 will find the global minimum of the objective function.

4.1 FS-1

In this variant we fix the scaling factor $\boldsymbol{\alpha} = \mathbf{1}$, thereby reducing the number of parameters to be tuned. However, this model cannot adjust for the different value range of features and can only learn the shiftings required. We name this variant as **FS-1** and is given by,

$$\sigma_i(x_i^n) = \frac{1}{1 + \exp(-x_i^n + \beta_i)}. \tag{14}$$

The update equations for w_j, b, and β_j are as follows,

$$w_j^{k+1} = w_j^k(1 - 2\lambda\eta_k) + \eta_k(t_n - y_n)\sigma_j(x_j^n), \tag{15}$$

$$b^{k+1} = b^k + \eta_k(t_n - y_n), \tag{16}$$

$$\beta_j^{k+1} = \beta_j^k(1 - 2\nu\eta_k) - \eta_k(t_n - y_n)w_j\sigma_j(x_j^n)(1 - \sigma_j(x_j^n)). \tag{17}$$

Note that although the update Equations 15, 16, and 17 appears to be similar in their form to Equations 8, 9, and 11, the transformation functions in the two sets of equations are different. As discussed earlier, **FS-1** is convex with respect to $\boldsymbol{w}$ and b, but non-convex with respect to $\boldsymbol{\beta}$.

4.2 FS-2

We design a convex form of the objective function with respect to all parameters by replacing the sigmoid feature scaling function with a linear combination as follows,

$$\sigma_i(x_i) = \alpha_i x_i + \beta_i. \tag{18}$$

The class conditional probability is computed using the logistic sigmoid model as,

$$P(t_n = 1|\boldsymbol{w}, b, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{1 + \exp(-\sum_{j=1}^{M} w_j(\alpha_j x_j^n + \beta_j) - b)} \tag{19}$$

Then the update equations for $\boldsymbol{w}$, b, $\boldsymbol{\alpha}$, and $\boldsymbol{\beta}$ are given as follows,

$$w_j^{k+1} = w_j^k(1 - 2\lambda\eta_k) - \eta_k(y_n - t_n)(\alpha_j x_j^n + \beta_j), \tag{20}$$

$$b^{k+1} = b^k - \eta_k(y_n - t_n), \tag{21}$$

$$\alpha_j^{k+1} = \alpha_j^k(1 - 2\mu\eta_k) - \eta_k(y_n - t_n)w_j x_j^n, \tag{22}$$

$$\beta_j^{k+1} = \beta_j^k(1 - 2\nu\eta_k) - \eta_k(y_n - t_n)w_j. \tag{23}$$

Here, we used $y_n = P(t_n = 1|\boldsymbol{w}, b, \boldsymbol{\alpha}, \boldsymbol{\beta})$ to simplify the equations.

Moreover, the second-order partial derivatives of the objective function E, with respect to $\boldsymbol{w}$, b, $\boldsymbol{\alpha}$, and $\boldsymbol{\beta}$ can be computed as follows,

$$\frac{\partial^2 E}{\partial w_j^2} = y_n(1 - y_n)(\alpha_j x_j^n + \beta_j)^2 + 2\lambda,$$

$$\frac{\partial^2 E}{\partial \alpha_j^2} = y_n(1 - y_n)w_j^2 x_j^{n2} + 2\mu,$$

$$\frac{\partial^2 E}{\partial \beta_j^2} = y_n(1 - y_n)w_j^2 x_j^{n2} + 2\mu,$$

$$\frac{\partial^2 E}{\partial w_j^2} = y_n(1 - y_n).$$

From, $0 < y_n < 1$, $\lambda > 0$, $\mu > 0$, and $\nu > 0$ it follows that all the above-mentioned second-order derivatives are positive, which proofs the convexity of the objective function. We name this convex formulation of the feature scaling method as the **FS-2** method.

4.3 FS-3

Although **FS-2** is convex, there is an issue regarding the determinability among $\boldsymbol{w}$, $\boldsymbol{\alpha}$, and $\boldsymbol{\beta}$ because the product between $\boldsymbol{w}$ and $\boldsymbol{\alpha}$, and the product between $\boldsymbol{w}$ and $\boldsymbol{\beta}$ appears inside the exponential term in Equation 19. This implies that the probability $P(t_n = 1|\boldsymbol{w}, b, \boldsymbol{\alpha}, \boldsymbol{\beta})$ will be invariant under a constant scaling of $\boldsymbol{w}$, $\boldsymbol{\alpha}$, and $\boldsymbol{\beta}$. We can absorb the w_j terms from the objective function into the corresponding α_j and β_j terms thereby effectively both reducing the number of parameters to be trained as well as eliminating the issue regarding the determinability. We name this variant of the feature scaling method as the **FS-3** method.

The class conditional probability for **FS-3** is give by,

$$P(t_n = 1|b, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{1 + \exp(-\sum_{j=1}^{M}(\alpha_j x_j^n + \beta_j) - b)}. \tag{24}$$

This can be seen as a special case of **FS-2** where we set $\boldsymbol{w} = \mathbf{1}$ and $\lambda = 0$.

The update equations for **FS-3** can be derived as follows,

$$b^{k+1} = b^k - \eta_k(y_n - t_n), \tag{25}$$

$$\alpha_j^{k+1} = \alpha_j^k(1 - 2\mu\eta_k) - \eta_k(y_n - t_n)x_j^n, \tag{26}$$

$$\beta_j^{k+1} = \beta_j^k(1 - 2\nu\eta_k) - \eta_k(y_n - t_n). \tag{27}$$

Here, we used $y_n = P(t_n = 1|b, \boldsymbol{\alpha}, \boldsymbol{\beta})$ to simplify the equations. Because **FS-2** is convex and **FS-3** is a special case of **FS-2**, it follows that **FS-3** is also convex.

5. Datasets

To evaluate the performance of the numerous feature scaling methods introduced in Section 4, we train and test those methods under the one-pass online learning setting. We use three datasets in our experiments: **heart** dataset, **liver** dataset, and the **diabetes** dataset. All three datasets are popularly used as benchmark datasets to evaluate binary classification algorithms. Moreover, all three datasets contain real-valued and unscaled features values, which suits our purpose. All three datasets can be downloaded from the UCI Machine Learning Repository[*1]. Details of the three datasets are summarized in Table 1.

[*1] `http://archive.ics.uci.edu/ml/`

Table 1 Statistics regarding the three datasets used in the experiments.

Dataset	Attributes	Train instances	Test instances
heart	13	216	54
liver	6	276	69
diabetes	8	611	157

6. Experiments and Results

We compare the performance of the following approaches on the three benchmark datasets.

SGD: This method implements logistic regression using stochastic gradient descent. It does not use any feature scaling and uses the original feature values as they are for training a binary classifier. This method demonstrates the lower baseline performance for this task.

SDG+avg: This method is same as **SGD** described above, except that it uses the average weight vector during training and testing. Specifically, it computes the average of the weight vector $\boldsymbol{w}$ over the updates and uses this average vector for prediction. By considering the average weight vector instead of the final weight vector we can avoid any bias toward the last few training instances encountered by the online learner. Moreover, it has been shown both theoretically and empirically that consideration of the average weight vector results in faster convergence in online learning [5].

GN: This is the unsupervised feature scaling method described in Section 3. It trains a binary logistic regression model by scaling the features using the unsupervised approach.

GN+avg: This is the unsupervised feature scaling method described in Section 3 using the average weight vector for predicting instead of the final weight vector. It trains a binary logistic regression model by scaling the features using the unsupervised approach.

FS: This is the method described in Section 4.

FS+avg: This is the **FS** method, where we use the average values for all parameters: $\boldsymbol{w}$, b, $\boldsymbol{\alpha}$, and $\boldsymbol{\beta}$.

FS-1: This is the method described in Section 4.1.

FS-1+avg: This is the method described in Section 4.1 with averaged parameter vectors.

FS-2: This is the method described in Section 4.2.

FS-2+avg: This is the method described in Section 4.2 with averaged parameter vectors.

FS-3: This is the method described in Section 4.3.

FS-3+avg: This is the method described in Section 4.3 with averaged parameter vectors.

PA: This is the Passive-Aggressive binary linear classification algorithm proposed in [6].

PA+avg: This is the Passive-Aggressive binary linear classification algorithm proposed in [6] using the averaged weight vector to predict during both training and testing stages.

PA-1: This is the Passive-Aggressive PA-I version of the binary linear classification algorithm proposed in [6].

PA-1+avg: This is the Passive-Aggressive PA-I version of the binary linear classification algorithm proposed in [6] using the averaged weight vector to predict during both training and testing stages.

PA-2: This is the Passive-Aggressive PA-II version of the binary linear classification algorithm proposed in [6].

PA-2+avg: This is the Passive-Aggressive PA-II version of the binary linear classification algorithm proposed in [6] using the averaged weight vector to predict during both training and testing stages.

We measure train and test error for each of the above-mentioned 18 algorithms using the three benchmark dataset, The experimental results are shown in Tables 2, 3, and 4. Error is computed as follows,

Table 2 Results on the heart dataset.

Algorithm	Train Error	Test Error	Best Parameters
SGD	0.537037	0.574074	$\lambda = 0.01$
SGD+avg	0.481481	0.435185	$\lambda = 0$
GN	**0.87037**	**0.824074**	$\lambda = 0.01$
GN+avg	0.777778	0.768519	$\lambda = 0.1$
FS	0.592593	0.49537	$\lambda = 0.1, \mu = 1.0, \nu = 0$
FS+avg	0.481481	0.435185	$\lambda = 0, \mu = 0\nu = 0$
FS-1	0.703704	0.564815	$\mu = 100.0, \nu = 0.1$
FS-1+avg	0.759259	0.564815	$\mu = 0.1, \nu = 10.0$
FS-2	0.740741	0.569444	$\lambda = 10.0, \mu = 0, \nu = 10.0$
FS-2+avg	0.574074	0.467593	$\lambda = 0, \mu = 1.0, \nu = 0$
FS-3	0.592593	0.476852	$\mu = 0.1, \nu = 0$
FS-3+avg	0.574074	0.421296	$\mu = 0.1, \nu = 1.0$
PA	0.648148	0.675926	$c = 0.01$
PA+avg	0.611111	0.662037	$c = 0.01$
PA1	0.648148	0.675926	$c = 0.01$
PA1+avg	0.611111	0.662037	$c = 0.01$
PA2	0.648148	0.675926	$c = 0.01$
PA2+avg	0.611111	0.662037	$c = 0.01$

Table 3 Results on the liver dataset.

Algorithm	Train Error	Test Error	Best Parameters
SGD	0.608696	0.561594	$\lambda = 0.1$
SGD+avg	0.550725	0.586957	$\lambda = 0$
GN	0.695652	0.637681	$\lambda = 100.0$
GN+avg	**0.777778**	**0.768519**	$\lambda = 0.1$
FS	0.637681	0.586957	$\lambda = 10.0, \mu = 0.1, \nu = 0.1$
FS+avg	0.550725	0.586957	$\lambda = 0, \mu = 0\nu = 0$
FS-1	0.623188	0.413043	$\mu = 1.0, \nu = 0$
FS-1+avg	0.623188	0.413043	$\mu = 0.1, \nu = 0.1$
FS-2	0.681159	0.59058	$\lambda = 0, \mu = 0.01, \nu = 0$
FS-2+avg	0.550725	0.586957	$\lambda = 0, \mu = 0, \nu = 0$
FS-3	0.623188	0.550725	$\mu = 0, \nu = 0$
FS-3+avg	0.550725	0.586957	$\mu = 0, \nu = 0$
PA	0.434783	0.427536	$c = 0.01$
PA+avg	0.565217	0.594203	$c = 0.01$
PA1	0.434783	0.427536	$c = 0.01$
PA1+avg	0.565217	0.594203	$c = 0.01$
PA2	0.434783	0.427536	$c = 0.01$
PA2+avg	0.565217	0.594203	$c = 0.01$

Table 4 Results on the diabetes dataset.

Algorithm	Train Error	Test Error	Best Parameters
SGD	0.643312	0.653028	$\lambda = 1.0$
SGD+avg	0.643312	0.653028	$\lambda = 0$
GN	0.656051	0.656301	$\lambda = 0.01$
GN+avg	**0.656051**	**0.671031**	$\lambda = 100.0$
FS	0.643312	0.653028	$\lambda = 0, \mu = 0, \nu = 0$
FS+avg	0.643312	0.653028	$\lambda = 0, \mu = 0\nu = 0$
FS-1	0.643312	0.653028	$\mu = 10, \nu = 0$
FS-1+avg	0.643312	0.653028	$\mu = 0, \nu = 0$
FS-2	0.643312	0.653028	$\lambda = 0, \mu = 0, \nu = 1.0$
FS-2+avg	0.643312	0.653028	$\lambda = 0, \mu = 0, \nu = 0$
FS-3	0.643312	0.653028	$\mu = 0.01, \nu = 100.0$
FS-3+avg	0.643312	0.653028	$\mu = 0, \nu = 0.01$
PA	0.611465	0.656301	$c = 0.01$
PA+avg	0.636943	0.657938	$c = 0.01$
PA1	0.648148	0.675926	$c = 0.01$
PA1+avg	0.636943	0.657938	$c = 0.01$
PA2	0.611465	0.656301	$c = 0.01$
PA2+avg	0.636943	0.657938	$c = 0.01$

$$\frac{\text{total no. of misclassified instances}}{\text{total no. of instances in the dataset}}. \tag{28}$$

We vary the parameter values for the numerous parameters in a pre-defined set of values for each parameter and experiment with all possible combinations of those values. For the regularization coefficients λ, μ, and ν we experiment with the values in the set $\{0, 0.01, 0.1, 1, 10, 100\}$. For the c parameter in passive-aggressive algorithms we chose from the set $\{0.01, 0.1, 1, 10, 100\}$. We show the results for the parameter values that yields the best test accuracy on each dataset. Therefore, the results reported in the paper can be considered as the best possible results that can be achieved with each method on the benchmark datasets. In a more conservative setting, in which one might select the parameter values using a separate development dataset independent from both the train and test datasets, the results on the test dataset might be lower.

Online learning algorithms have been shown to be sensitive to the order in which training examples are presented to them. Following the suggestions in prior work, we randomize the sequence of training data instances during training [2].

As can be seen from Tables 2, 3, and 4 the unsupervised feature scaling methods (**GN** and **GN+avg**) consistently outperform joint supervised feature scaling methods and PA algorithms. This result can be imply several possibilities. First, it could be that it is possible to accurately estimate the mean and standard deviation for each feature accurately even in the one-pass online learning setting for the three datasets used for evaluation. Although the estimates for mean and standard deviation of a particular feature might be incorrect at the initial stages of the learning, when we encounter more and more training instances we will be able to accurately estimate the mean and standrad deviation of a particular feature. It also helps for the unsupervised feature scaling method that all the three datasets used in the paper consist of dense feature vectors without any missing values. For example, if a particular feature only appears in a small subset of the training instances, then it would be difficult to accurately estimate the mean and the standard deviation of that feature, hence unable to scale that feature correctly using the unsupervised method. Therefore, we plan to repeat the experiments with feature sparse datasets such as ones that are frequently encountered in the text classification tasks. However, those features values are either binary (i.e. whether a particular word occurs in a particular document) or integer (i.e. the frequency of a particular word in a particular document) or a computed real-valued score from frequency counts such as tfidf [16]. values. Because feature scaling becomes an issue mostly with real-valued features, it is unknown whether document classification datasets, as used by most online learning algorithms for evaluation purposes, it suitable to demonstrate the effect of a feature scaling algorithm.

Among the variants of the proposed **FS** methods, the **FS-2** method reports the best performance. This can be attributable to the convexity of the objective function. Because we are allowed only a single pass over the training dataset in OPOL setting, convergence becomes a critical issue compared to the classical online learning setting where the learning algorithm traverses multiple times over the dataset. Therefore, the convexity of the objective

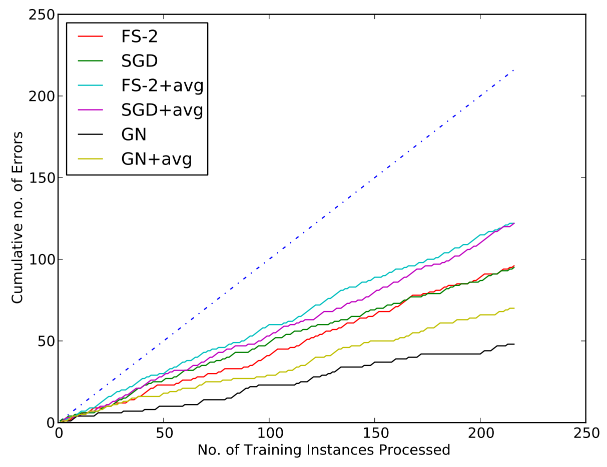

Fig. 1 Cumulative training errors on the heart dataset.

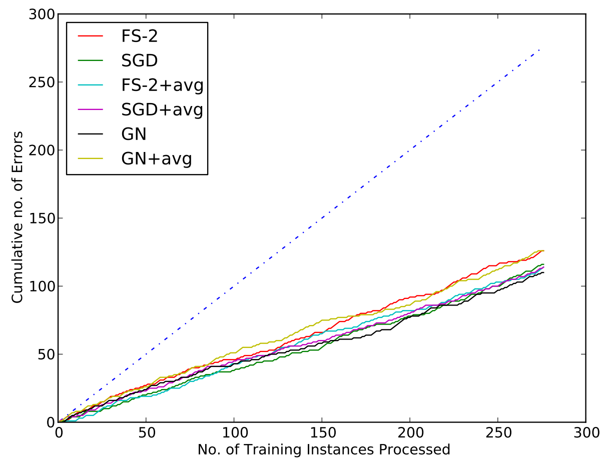

Fig. 2 Cumulative training errors on the liver dataset.

function might be an important factor that assists the process of convergence to a global optimal solution in a single pass over the dataset.

The difference between a particular learning algorithm and its averaged counterpart (**+avg** version) is mixed. On the liver dataset we see that the averaged version always outperforms the non-averaged version for the same algorithm. However, the trend is reversed on the heart dataset, whereas in the diabetes dataset there is not much difference between the two versions.

To study the behavior of the different learning algorithms during train time, we compute the cumulative number of errors. We plot the cumulative number of errors against the total number of training instances encountered as shown in Figures 1, 2, and 3 respectively for heart, liver, and diabetes datasets. During training, we use the weight vector (or the averaged weight vector for the **+avg** algorithms) to classify the current training instances and if it is misclassified by the current model, then it is counted as an error. The 45 degree line in each plot correspond to the situation where all instances encountered during training are misclassified. All algorithms must lie below this line. If the cumulative number of error curve for an algorithm becomes closer to the x-axis, it is

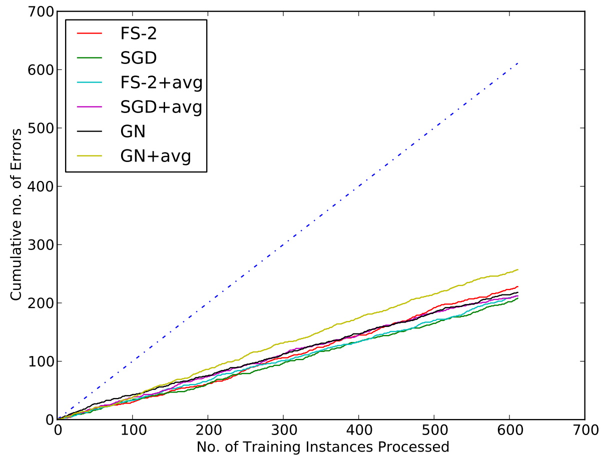

Fig. 3 Cumulative training errors on the diabetes dataset.

considered superior because it will be making lesser number of errors even during the train stage. To avoid cluttering, we only show the cumulative number of error curves for the following six methods: **FS-2**, **FS-2+avg**, **SGD**, **SGD+avg**, **GN**, and **GN+avg**. We see that **GN** method stands out among all the methods shown in Figure 1 in the heart dataset. The difference among the methods compared is less apparent for the other two datasets. **FS-2+avg** method shows the best cumulative number of errors on the diabetes dataset as can be seen from Figure 3.

7. Related Work

To our knowledge, the topic of feature scaling has not yet been studied in a supervised setting. However, the unsupervised feature scaling approach presented in Section 3 has been used both in batch as well as online learning settings. On the other hand, the previous literature on online learning is extensive and we cannot cover all in this Section. Some notable algorithms are the passive-aggressive algorithms [6], confidence-weighted linear [10] and multi-class classifiers [7], [8], [11], [13], [14], [15], active-learning [9] and transfer learning [17] approaches based on online learning.

The confidence-weighted approaches for online classifier learning are closest in spirit to our objective of feature scaling. In confidence-weighted learning, a Gaussian prior is imposed on each weight of a feature. During training, these algorithms not only learn the value (corresponds to the mean) of the weight of a feature, but also its confidence (standard deviation). Full covariance matrices that can account for the inter-feature relations as well as a diagonal approximations of the feature covariance matrix have been studied in previous work. However, the relationship between feature scaling and confidence-weighted learning is not made explicit thus far.

8. Conclusion

We studied the problem of feature scaling in one-pass online learning of binary linear classifiers. We presented both supervised as well as unsupervised approaches for feature scaling and evaluated 18 different learning methods using three popular datasets. Our experimental results show that the unsupervised approach clearly outperformed the supervised approaches on all three datasets. Among the several variants of the supervised feature scaling approach we evaluated, the convex formulation performed best. In future, we plan to explore other forms of feature scaling functions and evaluate on larger real-valued sparse datasets.

References

[1] Bengio, Y., Louradour, J., Collobert, R. and Weston, J.: Curriculum Learning, *ICML'09*, pp. 41 – 48 (2009).
[2] Bertsekas, D. P.: Incremental Gradient, Subgradient, and Proximal Methods for Convex Optimization: A Survey, Technical Report LIDS 2848, MIT (2010).
[3] Bishop, C. M.: *Pattern Recognition and Machine Learning*, Springer (2006).
[4] Chan, T. F., Golub, G. H. and LeVeque, R. J.: Algorithms for Computing the Sample Variance: Analysis and Recommendations, *The American Statistician*, Vol. 37, pp. 242 – 247 (1983).
[5] Collins, M.: Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms, *EMNLP'02*, pp. 1 – 8 (2002).
[6] Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S. and Singer, Y.: Online Passive-Aggressive Algorithms, *Journal of Machine Learning Research*, Vol. 7, pp. 551–585 (2006).
[7] Crammer, K., Dredze, M. and Kulesza, A.: Multi-Class Confidence Weighted Algorithms, *EMNLP 2009*, pp. 496 – 504 (2009).
[8] Crammer, K., Dredze, M. and Pereira, F.: Exact Convex Confidence-Weighted Learning, *NIPS'08* (2008).
[9] Dredze, M. and Crammer, K.: Active Learning with Confidence, *ACL'08*, pp. 233 – 236 (2008).
[10] Dredze, M., Crammer, K. and Pereira, F.: Confidence-Weighted Linear Classification, *ICML'08*, pp. 264 – 271 (2008).
[11] Duchi, J., Hazan, E. and Singer, Y.: Adaptive Subgradient Methods for Online Learning and Stochastic Optimization, *COLT'10* (2010).
[12] Ling, R. F.: Comparison of Several Algorithms for Computing Sample Means and Variances, *Journal of the American Statistical Association*, Vol. 69, No. 348, pp. 859 – 866 (1974).
[13] Ma, J., Kulesza, A., Dredze, M., Crammer, K., Saul, L. K. and Pereira, F.: Exploiting Feature Covariance in High-Dimensional Online Learning, *AISTAT'10* (2010).
[14] Mejer, A. and Crammer, K.: Confidence in Structured-Prediction using Confidence-Weighted Models, *EMNLP'10*, pp. 971 – 981 (2010).
[15] Mejer, A. and Crammer, K.: Confidence Estimation in Structured Prediction, Technical Report 798, Center for Communication and Information Technologies (2011).
[16] Salton, G. and Buckley, C.: *Introduction to Modern Information Retreival*, McGraw-Hill Book Company (1983).
[17] Zhao, P. and Hoi, S. C. H.: OTL: A Framework of Online Transfer Learning, *ICML 2011* (2011).