Cost-sensitive Learning for Large-scale Hierarchical Classification of Commercial Products

Jianfu Chen
Computer Science Department
Stony Brook University
Stony Brook, NY 11790
jianchen@cs.stonybrook.edu

David Warren
Computer Science Department
Stony Brook University
Stony Brook, NY 11790
warren@cs.stonybrook.edu

ABSTRACT

We study hierarchical classification of products in electronic commerce, classifying a text description of a product into one of the leaf classes of a tree-structure taxonomy. In particular, we investigate two essential problems, *performance evaluation* and *learning*, in a synergistic way. Unless we know what is the appropriate performance evaluation metric for a task, we are not going to learn a classifier of maximum utility for the task. Given the characteristics of the task of hierarchical product classification, we shed insight into ***how and why*** common evaluation metrics such as error rate can be misleading, which is applicable for treating other real world applications. The analysis leads to a new performance evaluation metric that tailors this task to reflect a vendor's business goal of maximizing revenue. The proposed metric has an intuitive meaning as the average revenue loss, which depends on *both* the **value** of individual products *and* the hierarchical **distance** between the true class and the predicted class. Correspondingly, our learning algorithm uses multi-class SVM with margin re-scaling to optimize the proposed metric, instead of error rate or other common metrics. Margin re-scaling is sensitive to the scaling of loss functions. We propose a **loss normalization** approach to appropriately calibrating the scaling of loss functions, which is applicable to general classification and structured prediction tasks whenever using structured SVM with margin re-scaling. Experiments on a large dataset show that our approach outperforms standard multi-class SVM in terms of the proposed metric, effectively reducing the average revenue loss.

Categories and Subject Descriptors

I.5.2 [**Pattern Recognition**]: Design Methodology—*Classifier design and evaluation*; I.2.6 [**Artificial Intelligence**]: Learning—*Induction*; H.3.3 [**Information Storage and Retrieval**]: Information Search and retrieval—*Information Filtering*

Keywords

product classification; cost-sensitive learning; taxonomy; hierarchical classification; SVM; margin re-scaling; loss normalization; document classification; revenue loss; UNSPSC

1. INTRODUCTION

E-commerce enables customers to buy products any time and anywhere. E-commerce has expanded rapidly over the last decade, and is predicted to continue its fast growth with the rise of smartphones and tablets.

In an online shopping platform, it is vital to enable customers find desired products quickly. To this end, the two most popular mechanisms are taxonomy organization and keyword search. A taxonomy organizes products by categories grouped hierarchically in a tree structure, from general classes to more specific classes. Two examples are the catalogs of Amazon.com and eBay.com. Such taxonomies complement keyword search. While keyword search is good for quickly finding a very specific product in a customer's mind, a taxonomy organization also has its merits: (1) *facilitates exploring similar products.* Categories are like departments in a supermarket that allow a customer to navigate and roam around a vast number of aisles. Are you looking for gift ideas for a kid? Just browse the sub-categories under the general "Toys&Games" category in Amazon.com. You will find many possibilities in a systematic way, e.g., dolls, puzzles, and building toys. In fact, even for a keyword query, many online shopping websites allow a customer to browse the search results by categories organized in a taxonomy. This helps a customer to filter out the really relevant categories. (2) *helps product recommendation.* Intuitively, given the products previously browsed or bought by a customer, we can use a taxonomy to recommend similar products in similar categories. Formally, [27] studies exploiting large taxonomies for personalized product recommendation.

Given the merits of a taxonomic organization of products, we explore automatic classification of textual product descriptions into a given taxonomy. Assume we are given a taxonomy that is a tree structure, where each product belongs to a single leaf class, and therefore also belongs to its more general ancestor classes. We want to classify a textual description of a product into one of the leaf nodes of a taxonomy. Figure 1 shows part of a product taxonomy called UNSPSC [1] (see § 4.1.1 for more details).

In particular, we investigate two essential problems, *performance evaluation* and *learning*, in a synergistic way. Unless we know what is the appropriate performance evaluation metric for a task, we are not going to learn a classifier that

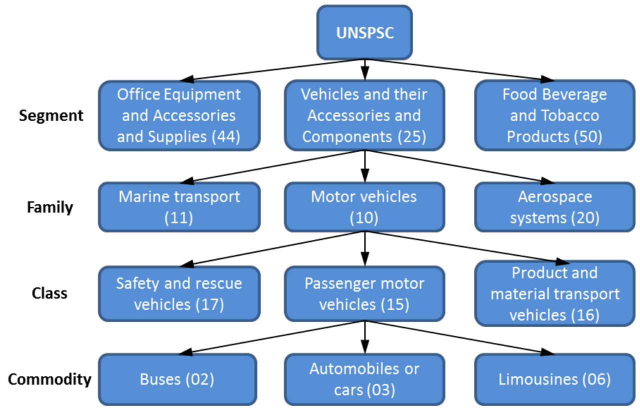
UNSPSC
Segment
Office Equipment and Accessories and Supplies (44)
Vehicles and their Accessories and Components (25)
Food Beverage and Tobacco Products (50)
Family
Marine transport (11)
Motor vehicles (10)
Aerospace systems (20)
Class
Safety and rescue vehicles (17)
Passenger motor vehicles (15)
Product and material transport vehicles (16)
Commodity
Buses (02)
Automobiles or cars (03)
Limousines (06)

Figure 1: Part of the UNSPSC taxonomy

has maximum utility for the task. We study them under a unified view of empirical risk [24], the average loss or misclassification cost. A performance evaluation metric defines a type of misclassification cost. Learning optimizes the average misclassification cost.

Performance evaluation is a seemingly trivial problem, and hence is often neglected in real world applications. However, we argue that we should choose an appropriate performance evaluation metric according to the characteristics of the task, rather than just blindly choosing a common evaluation metric like error rate. We examine the special characteristics of the task of hierarchical product classification, where a vendor classifies products with a business goal of maximizing revenue. We shed insight into ***how and why*** common evaluation metrics can be misleading. The analysis covers metrics including error rate, mean error rate, average hierarchical loss, and average F1-score, which is applicable when considering evaluating other real world tasks. Then we design a new evaluation metric that fixes the problems of common evaluation metrics and tailors this task to reflect a vendor's business goal of maximizing revenue. The proposed metric is essentially the average revenue loss, which depends on *both* the potential **revenue** of individual products *and* the hierarchical **distance** of the true class and the predicted class in the taxonomy.

After choosing an appropriate performance evaluation metric for the task of hierarchical product classification, we explore learning a classifier that optimizes the proposed evaluation metric, average revenue loss, rather than error rate as commonly done by standard classifiers. We use a generalization of multi-class SVM with margin re-scaling [23, 7] to optimize *any loss functions*. It is a general approach to handle cost-sensitive learning. However, margin re-scaling is sensitive to the scaling of loss functions, especially when the loss function, revenue loss, can span a wide range. We propose an approach to normalize the loss function into a fixed range, appropriately calibrating the scaling of the loss functions. Our **loss normalization** approach is applicable to other classification and structured prediction tasks, whenever using structured SVM with margin re-scaling.

Finally, we perform experiments on a large dataset that has more than one million products in about one thousand leaf classes. The results show that our approach outperforms standard multi-class SVM in terms of our proposed evaluation metric, significantly reducing the average revenue loss.

Our work is an application of cost-sensitive learning when different misclassification have different costs [11, 9, 26, 25]. Very few works [3] study *both* example-dependent cost *and* class-dependent cost, especially in a practical scenario as we do. Though we study a particular task, this task represents an emerging class of applications that involve both a taxonomy and items with individual values in large scale information management.

2. PERFORMANCE EVALUATION FOR HIERARCHICAL PRODUCT CLASSIFICATION

In this section, we first dissect the characteristics of the task of hierarchical product classification and envision the properties of a desirable performance evaluation metric. Then under a unified view of empirical risk, we analyze that common performance evaluation metrics, including *accuracy, mean accuracy, and average hierarchical loss,* fail to reflect a vendor's business goal sufficiently. Finally we propose a new evaluation metric that fixes the problems of common evaluation metrics.

2.1 Characteristics of the task

A close look at the uses of the classified products in the task motivates us to delve into the issue of classification performance evaluation.

2.1.1 Business goal

Consider the application scenario of product classification. A vendor or an online shopping platform classifies a set of products into the leaf classes of a predefined taxonomy. The vendor wants to classify the products as accurately as possible, so that potential customers can explore and find their needed products without obstacles. *We assume that if a product is classified into the correct class, the vendor will realize an expected annual revenue from that product. Otherwise, the vendor will lose some potential revenue, realizing only part of the expected annual revenue of that product, because customers have trouble finding and buying that product.* Hereafter all references to revenue are meant to be calculated within one year. Hence we drop the modifier "annual" for better readability.

A vendor's business goal is to **maximize revenue**. How should we evaluate a classifier's performance? To tailor a vendor's interest, a reasonable measure is the revenue loss caused by the classifier's misclassification.

The next question is: how much revenue will a vendor lose due to the misclassification of a product into a wrong class? Before we quantify the revenue loss in our proposed evaluation metric in § 2.4, we first do some qualitative comparative analysis in the next section.

2.1.2 Taxonomy organization and customer behavior assumption

A qualitative analysis shows *the misclassification cost, or equivalently, revenue loss of a product into a sibling class should be smaller than that into a far-away class in the product taxonomy.* The analysis is based on the properties of the taxonomy organization and an assumption about the customer behavior in online shopping.

A product taxonomy groups product classes hierarchically in a tree structure, from general classes to more specific classes. Classes that share a common parent class are similar to each other. For example, the class "*mouse*" and the class "*computer keyboard*" share a common parent class "*desktop computer and accessories*"; the classes "*hard drive*", "*SSD*", and "*USB*" share a common parent class "*computer storage*". Those child classes are similar in the sense that they have similar functions, or dominant usages in the market.

We assume when a customer browses products by categories, the customer frequently looks at sibling classes. Sibling classes have similar or related products. A customer often compares similar products before buying them. One often wonders between the choice of "*hard drive*" products and "*SSD*" products. A customer also frequently purchases related products, say "*mouse*" and "*computer keyboard*" together.

Given the above properties of a product taxonomy and the assumption about customer behavior, the amount of revenue loss incurred by the misclassification of a product into different false classes should be different. In particular, we differentiate the misclassification cost into a sibling class and that into a far-away class in terms of **hierarchical distance** defined as the length of the shortest path between two nodes. We have assumed that it is highly likely that a customer looks at sibling classes when browsing products by categories. Suppose one product in the *keyboard* class, is misclassified into a sibling class, say *mouse*, even though a customer looking for *keyboard* products cannot find that product in its true class, the customer will likely check the sibling *mouse* class, hence can still find and buy that *keyboard* product. However, if that *keyboard* product is misclassified into a far-away class, say *car*, it is less likely that a customer will find and buy that *keyboard* product, because it is less likely that the customer will browse that far-away class *car* together with the true class *keyboard.* Hence the misclassification cost of a product into a sibling class should be smaller than that into a far-away class.

Given the characteristics of the task, now we turn to treat the problem of performance evaluation formally.

2.2 A unified view of classification performance evaluation

A unified view of a classification performance evaluation metric is *empirical risk*, which is the average loss incurred by classifying an example. This view helps us *both* to analyze performance evaluation metrics in § 2.3 and § 2.4, *and* to treat learning as empirical risk minimization in § 3.2.

Suppose we are given a set of labeled examples: $\{(x, y)\}$, where each example $x \in R^N$ represents a product; x belongs to class $y \in Y$, where Y is the set of the leaf classes in the product taxonomy. Assume each labeled example (x, y) is drawn *i.i.d.* from an underlying joint distribution $P(x, y)$. A classifier is a function $f(x)$ that maps a given example x onto a class label $y' \in Y$.

A unified view of common classifier performance evaluation metrics is **empirical risk** [24] , which is the average loss incurred by classifying an example. Formally, let the loss function $L(x, y, y')$ represents the loss incurred by classifying an example x that belongs to class y into class y'; the empirical risk is the average loss of the classification over all examples,

$$R_{em} = \frac{1}{m} \sum_{(x,y,y') \in D} L(x, y, y')$$

where m is the total number of examples (m *represents the same meaning hereafter*). Typically, for correct classification, $L(x, y, y') = 0$; otherwise, $L(x, y, y') > 0$. So the lower the empirical risk is, the better the classifier performs.

In particular, we consider a class of loss functions that can be factorized as *a weighted classification error*,

$$L(x, y, y') = w(x) \cdot \triangle(y, y')$$

The **error function** $\triangle(y, y')$ quantifies the error of classifying an example that belongs to class y into class y'. The error function depends on only the true class y and the predicted class y'. Typically, for correct classification, $\triangle(y, y') = 0$; otherwise, $\triangle(y, y') > 0$. The **example weight** $w(x) \geq 0$ represents the importance of the example x's error function.

With such loss functions, we interpret empirical risk as **a weighted sum of classification errors**, ignoring the constant term $\frac{1}{m}$,

$$\frac{1}{m} \sum_{(x,y,y') \in D} w(x) \cdot \triangle(y, y') \tag{1}$$

The higher the example weight $w(x)$ is, the more importance the error function $\triangle(y, y')$ associated with example x has in the weighted sum.

2.3 The problems of common performance evaluation metrics

Based on the characteristics of the task of hierarchical product classification, we analyze that common classification performance evaluation metrics, including *error rate*, *mean error rate*, *average hierarchical loss*, and *average F1-score,* do not adequately reflect a vendor's business goal of maximizing revenue.

2.3.1 Error rate

The simplest loss function is a boolean function $L(x, y, y') = [y \neq y']$, where $[\cdot]$ is the Iverson bracket that returns the $0/1$ boolean value of the inside condition. If the example is misclassified, then the loss is 1; otherwise the loss is 0. Hence it is called *0-1 loss.* The empirical risk as the average 0-1 loss over the set D becomes,

$$\frac{1}{m} \sum_{(x,y,y') \in D} [y \neq y'] \tag{2}$$

This is equivalent to the number of misclassified examples divides by the total number of examples. So the empirical risk with 0-1 loss is called **error rate**.

We interpret Eq.(2) as a special case of a weighted sum of classification errors in Eq.(1), where the example weight $w(x) = 1$, and error function $\triangle(y, y') = [y \neq y']$. We see that error rate has two problems that render it inadequate to reflect a vendor's business goal of maximizing revenue:

(1) *It gives an equal weight to each example.* A direct consequence is that error rate favors the performance in large classes that have many examples relative to other classes. This problem becomes severe when the class distribution is highly skewed, as is the case in our task of product classification. Error rate might neglect the performance on small classes that have relatively very few examples.

To see this formally, we group the error functions of examples by classes, and rewrite error rate as *a weighted sum* *of the error rates in all classes.* Let the number of classes be K, and the size of class y, that is the number of examples in class y, be S_y; we have,

$$\begin{aligned}\frac{1}{m}\sum_{(x,y,y')\in D}[y\neq y'] &= \frac{1}{m}\sum_{y=1}^{K}\sum_{(x,y,y')\in D}[y\neq y'] \\ &= \sum_{y=1}^{K}\frac{S_y}{m}\cdot\left\{\frac{\sum_{(x,y,y')\in D}[y\neq y']}{S_y}\right\} \\ &= \sum_{y=1}^{K}\frac{S_y}{m}\cdot Err_y\end{aligned}$$

where Err_y denotes the error rate in class y.

We see that error rate is a weighted sum of the error rates in individual classes, where the weight of each class is proportional to its class size. With a highly skewed class distribution, error rate is dominated by the error rates in large classes, while ignores the performance in small classes.

In product classification with a vendor's business goal of maximizing revenue, we should give importance to a class based on the total revenue of the products in that class. The higher the total revenue of a class relative to other classes, the larger weight we should give to that class, emphasizing the importance to classify most of the products in that class correctly. It is inappropriate to give high importance to a class merely because the class has a relatively large number of products.

(2) The second problem of error rate is that *it treats the misclassification cost of an example of class y into all other classes y' equally.* As long as an example is misclassified, the error function $[y \neq y']$ will be 1. However, in hierarchical product classification, we want to discriminate the misclassification cost of a product into different false classes. According to the properties of a product taxonomy and the assumption about consumer behavior, we have shown that the revenue loss of a product into sibling classes should be smaller than that into far-away classes in the taxonomy in § 2.1.2. If we cannot classify a product into the true class exactly, we want to classify it into a class as close as possible, to minimize the revenue loss.

In the next two sections, another two common performance evaluation metrics, *mean error rate* and *average hierarchical loss* address the above two problems, respectively.

2.3.2 Mean error rate

We have shown the first problem of error rate is that it gives an equal weight to each example, therefore favoring the performance in large classes, while neglecting the performance in small classes, when the class distribution is highly skewed. A straightforward remedy is to give equal weights to the error rates in individual classes. This is called *balanced error rate* [5]. We call it mean error rate here, since it is the average error rate over individual classes. This view enables us to generalize it to use different weighting methods later in this section. Formally, we define **mean error rate** as,

$$\sum_{y=1}^{K}\frac{1}{K}\cdot Err_y \tag{3}$$

where K is the number of classes, Err_y is the error rate in class y, the number of misclassified examples in class y divided by the total number of examples in class y; formally $Err_y = \frac{\sum_{(x,y,y')\in D}[y\neq y']}{S_y}$. We rewrite mean error rate to the equivalent form as a weighted sum of classification errors,

$$\frac{1}{m}\sum_{(x,y,y')}\frac{m}{KS_y}\cdot[y\neq y']$$

We see that mean error rate gives an equal weight $\frac{m}{KS_y}$ to each example in class y, such that the sum of the weights in any class is uniformly $\frac{m}{K}$.

Generalized mean error rate. A slight generalization of mean error rate is to give non-uniform weights to the error rates in individual classes in Eq.(3). As long as those weights are non-negative and sum to 1. We define **generalized mean error rate** as,

$$\sum_{y=1}^{K} w_y\cdot Err_y \tag{4}$$

where class weight $w_y \geq 0$ for any class y, subject to $\sum_{y=1}^{K} w_y = 1$; and Err_y is the error rate in class y, defined the same as in Eq.(3). Similarly to mean error rate above, we rewrite generalized mean error rate as a weighted sum of 0-1 error functions,

$$\frac{1}{m}\sum_{(x,y,y')}\frac{mw_y}{S_y}\cdot[y\neq y'] \tag{5}$$

We see that generalized mean error rate gives an equal weight $\frac{mw_y}{S_y}$ to each example in class y, such that the total weight of the examples in class y is mw_y. It is easy to see that **both error rate and mean error rate are special cases of generalized mean error rate, with different class priors, or equivalently class weights.**

How should we choose the class weights w_y's? Consider the first problem of error rate. With a highly skewed class distribution, there are huge differences among the weights of the large classes and those of the small classes. To alleviate this problem, a heuristic way is to let the weight of a class y be proportional to $\log(S_y)$, or similarly $\sqrt{S_y}$, where S_y is the number of examples in class y. In product classification, a more reasonable way is to set the importance of a class proportional to the total revenue of that class.

We have seen how mean error rate and its generalized version try to fix the first problem of error rate of giving an equal weight to every example. Nevertheless, from Eq.(5), we see that both approaches still give equal weights to examples in the same class. However, in product classification, even within the same class, the revenue of different examples can span a wide range. To reflect a vendor's business goal of maximizing revenue, a more reasonable approach is to set the weight of each example proportional to its revenue. We will do this in our proposed metric in § 2.4.

2.3.3 Average hierarchical loss

To fix the second problem of error rate of treating the misclassification cost into all false classes equally, a simple approach is to replace the 0-1 error function $\triangle(y,y') = [y\neq y']$ in the weighted sum of classification errors as Eq.(1) with an error function that differentiates the misclassification errors into different false classes. In particular, we use a loss matrix $L \in R^{K\times K}$ whose entry $L_{yy'}$ specifies the misclassification cost of an example from true class y to predicted class

y'. Let the example weight $w(x) = 1$, and error function in Eq.(1) $\triangle(y, y') = L_{yy'}$. We define **average hierarchical loss** as,

$$\frac{1}{m} \sum_{(x,y,y')} L_{yy'}$$

In the case of hierarchical classification, we let $L_{yy'} = f(d(y, y'))$, a monotonically non-decreasing function of the hierarchical distance of y and y' in the taxonomy.

However, this evaluation metric still gives equal weight to each example's error function, leaving the first problem of error rate untouched. Combining both the ideas of example weighting and differentiating the misclassification cost of an example into different classes, we will propose in § 2.4 a new evaluation metric that fixes both problems of error rate.

2.3.4 *Average F1-score*

For completeness, we analyze the problem of another common evaluation metric, **average F1-score** [21]. It is a *not* a linear function of loss functions on individual examples, hence cannot be naturally cast as a weighted sum of classification errors as the above evaluation metrics. There are two types of average F1-score: *micro-averaged* F1-score and *macro-averaged* F1-score. In a multi-class problem, micro-averaged F1-score is equivalent to accuracy, that is 1 minus error rate, and thus has the same problems as error rate. Macro-averaged F1-score is the average of the F1-score over all classes. The problem of macro-averaged F1-score is that it gives an equal weight to the performance of each class. In product classification, it makes more sense to give importance weight to each class based on the total revenues of the products in that class.

2.4 Proposed performance evaluation metric - average revenue loss

Combining both the ideas of example weighting and differentiating the misclassification cost into different false classes, we propose a new evaluation metric. Intuitively, the proposed metric represents the average revenue loss incurred by the classification over the products, therefore directly reflecting a vendor's business goal of maximizing revenue, or equivalently, minimizing revenue loss.

The proposed metric basically quantifies the assumptions we make in § 2.1. We assume that,

(1) If a product x is classified into the correct class, then its potential customers will be able to find it without hindrance, hence the vendor will *fully* realize a potential revenue $v(x)$ of product x.

(2) If a product x is misclassified into a wrong class, then its potential customers will have trouble to find and buy it, hence the vendor will lose some *percentage* of the potential revenue $v(x)$ of that product, depending on the hierarchical distance of the true class and the predicted class. The further apart the two classes are, the larger the percentage of the revenue the vendor will lose. We call such a percentage as a **loss ratio**. Formally, assume we are given the revenue loss ratio $L_{yy'}$ of classifying any product x of class y to class y', such that $0 \leq L_{yy'} \leq 1$, and $L_{yy'}$ is a monotonically non-decreasing function of the hierarchical distance $d(y, y')$ between y and y'. In particular, $L_{yy} = 0$ for any class y. In a general sense, a loss ratio gives a partial credit to the classification. Going up to the lowest common ancestor of classes y and y', we have a class that is correct in a coarser grain sense.

Based on the above two assumptions, the *revenue loss* of classifying a product x that belongs to class y into class y' becomes $v(x)\cdot L_{yy'}$. Let the loss function be the revenue loss, $L(x, y, y') = v(x)\cdot L_{yy'}$, we propose a performance evaluation metric to represent the **average revenue loss** caused by the classification over the products in the considered set,

$$\frac{1}{m} \sum_{(x,y,y')\in D} v(x) \cdot L_{yy'}$$

Technically, like other common evaluation metrics discussed, this evaluation metric is a weighted sum of classification errors in Eq.(1). It gives an importance weight $v(x)$ to each product that is proportional to the product revenue. It discriminates the misclassification cost of a product into different false classes based on the hierarchical distance of the true class and the false class. Therefore, it solves both the two problems of error rate in § 2.3.1. Moreover, it encompasses both error rate and average hierarchical loss as special cases, by setting $L_{yy'} = 1$ and $v(x) = 1$, respectively.

3. COST-SENSITIVE LEARNING FOR HIERARCHICAL PRODUCT CLASSIFICATION

After choosing the *average revenue loss* as the appropriate performance evaluation metric for this task of hierarchical product classification, we consider learning a classifier that performs best with respect to this metric.

Standard classifier learning techniques like SVM [7] usually try to optimize error rate by minimizing a convex upper bound of the error rate. Therefore, standard classifiers are expected to be optimal with respect to error rate, while is *not* necessarily optimal in terms of our proposed evaluation metric.

We propose a cost-sensitive learning algorithm to optimize the average revenue loss in the training set. The algorithm is based on multi-class SVM with margin re-scaling. It is a general approach for optimizing any misclassification cost. However, margin re-scaling is sensitive to the scaling of loss functions. We propose a **loss normalization** approach to make margin re-scaling achieve good performance in practice. The loss normalization approach is applicable to other classification and structured prediction tasks whenever using structured SVM with margin re-scaling.

3.1 Linear classifiers

We consider linear classifiers that have been shown to achieve state-of-the-art performance for document classification. Given an example $x \in R^N$, a linear classifier uses a weight vector $\theta_y \in R^N$ to score each class $y \in Y$ by the inner product $\theta_y^T x$; then predict the class y' with the highest score,

$$y' = f(x) = \arg\max_y \theta_y^T x$$

where θ_y's are parameters of the classifier. We use θ to represent the collection of all θ_y's.

3.2 Minimize empirical risk

Corresponding to that many performance evaluation metrics can be formulated as empirical risk in § 2.2, many learning algorithms can be formulated as minimizing empirical risk in the training set, additionally with parameter regularization [22]. Assume the parameters of a linear classifier are θ, learning becomes an optimization problem that finds the optimal parameter θ^* that minimizes the empirical risk $R_{em}(D;\theta)$, the average loss in the training set, with parameter regularization to avoid over fitting,

$$\begin{aligned}\theta^* &= \arg\max_{\theta} \frac{1}{2}||\theta||^2 + R_{em}(D;\theta) \\ &= \arg\max_{\theta} \frac{1}{2}||\theta||^2 + \frac{1}{m}\sum_{(x,y,y')\in D} L(x,y,y';\theta) \quad (6)\end{aligned}$$

where $L(x,y,y';\theta)$ is the loss function for an example x of class y and predicted into class y', given the classifier parameters θ.

3.3 Minimize average revenue loss

To learn the parameters of a linear classifier, we use multi-class SVM with margin re-scaling that scales the required margin according to the loss function, the revenue loss in our case. We also propose a loss normalization approach to make margin-rescaling work well in practice.

3.3.1 Margin re-scaling

Standard multi-class SVM by Crammer and Singer [7] indirectly optimizes error rate by minimizing a convex upper bound of average 0-1 loss. Similarly, we try to optimize average revenue loss by minimizing a convex upper bound.

Let the loss function in Eq.(6) be revenue loss, $L(x,y,y';\theta) = v(x)\cdot L_{yy'}$, then we optimize the average revenue loss directly. Unfortunately, this is a non-convex objective function, which is difficult to solve.

Similar to the *hinge loss* in multi-class SVM [7], we use a convex surrogate of the loss function,

$$\max\left\{0,\ \max_{y'\neq y}\left\{L(x,y,y';\theta)+\theta_{y'}^T x-\theta_y^T x\right\}\right\} \quad (7)$$

For correct classification, we want the score $\theta_y^T x$ of the *true* class y to be higher than the score $\theta_{y'}^T x$ of any *false* class y'. The larger the difference, the more confident our prediction is. In particular, we require the difference to be greater than the the misclassification cost of x from class y to class y': $\theta_y^T x - \theta_{y'}^T x \geq L(x,y,y';\theta) = v(x)\cdot L_{yy'}$, which is called a *margin* constraint; otherwise, we pay a positive loss for the margin violation. It is easy to prove that this is a convex upper bound of the loss function $L(x,y,y';\theta)$.

However, Eq.(7) is still non-differentiable. We reformulate the problem as a constrained optimization problem with parameter regularization similarly to [7]. To absorb the violations of the margin constraints, we use a slack variables ξ_i for each example. Let (x_i,y_i) be the ith training example, we have the following constrained optimization problem,

$$\begin{aligned}&\min_{\theta}\frac{1}{2}||\theta||^2+\frac{C}{m}\sum_{i=1}^{m}\xi_i \\ s.t.\ \forall i,\ \forall y'\neq y_i:\ \theta_{y_i}^T x_i-\theta_{y'}^T x_i &\geq L(x_i,y_i,y')-\xi_i \quad (8)\\ \xi_i &\geq 0\end{aligned}$$

where C is the regularization hyper parameter, $L(x_i,y_i,y') = v(x_i)L_{y_iy'}$ is the loss function. It is not difficult to prove that the solution to the above problem is also the solution to the unconstrained optimization problem as Eq.(6) with the hinge loss Eq.(7). So $\xi_i \geq L(x_i,y_i,y')$ for any y'; the objective is an upper bound of the empirical risk $\frac{1}{m}\sum_i L(x_i,y_i,y_i')$ (up to a constant factor C).

Eq.(8) is a special case of structured SVM with **margin-rescaling** [23]. It is also a generalization of multi-class SVM [7], which uses 0-1 loss $L(x_i,y_i,y') = [y_i \neq y']$ in Eq.(8).

Multi-class SVM with margin re-scaling shows a general way to do cost-sensitive learning, by scaling the margin proportional to any loss function, hence minimizing the upper bound of the average loss, empirical risk. In our experiments, we compare margin re-scaling by our proposed loss function (denoted as **REVLOSS**) to by three other loss functions, (1) the **0-1** loss, $L(x_i,y_i,y') = [y_i \neq y']$, giving the standard multi-class SVM; (2) the **VALUE** loss, that is the revenue of the product $L(x_i,y_i,y') = v(x_i)[y_i \neq y']$. This assumes misclassification causes losing the whole potential revenue. The misclassification cost does not depend on hierarchical distance of the true class and the predicted class. (2) the **TREE** loss, the height of the lowest common ancestor of the true class and the predicted class $L(x_i,y_i,y') = L_{y_iy'}$, with $L_{yy} = 0$. The misclassification cost does not depend on the product revenue.

3.3.2 Loss normalization

As §2.2.5 in [23] pointed out, margin re-scaling is sensitive to the scaling of the loss function. One should be careful in calibrating the scaling of the loss function with respect to the scaling of the feature values.

Indeed, in our experiments, margin re-scaling using the original revenue values in dollars performs significantly worse than standard multi-class SVM, even after rescaling the revenue by different units, say million \$. Because the annual revenue of individual products span from the order of hundred dollars to million dollars, hence so are the revenue loss. Therefore the required margin for high revenue products can be much larger than those for low revenue products. This means either (1) the required margin for large revenue products are too large, but the feature values are word frequencies most of which are 1 to 2, which tends to force the norm of the parameters to be large to reach large margin; or (2) the required margin for small revenue products are too small, close to zero, which essentially don't apply margin constraints for them. Both cases lead to larger generalization error.

To adjust the difference between the influence of extreme high revenue loss and that of extreme low revenue loss, we propose to linearly scale the loss function to a fixed range $[M_{min}, M_{max}]$,

$$L^s(x,y,y') = M_{min} + \frac{L(x,y,y')-L_{min}}{L_{max}-L_{min}}\cdot(M_{max}-M_{min})$$

where $L_{min} = \min\{L(x,y,y')\}$ is the minimum possible revenue loss in the training set, calculated as the product of the minimum revenue value times the minimum loss ratio; and similarly $L_{max} = \max\{L(x,y,y')\}$ is the maximum possible revenue loss. So that L_{min} is mapped to M_{min}, and L_{max} is mapped to M_{max}.

It is not hard to see that minimizing the empirical risk with normalized loss is equivalent to minimizing the original empirical risk in terms of choosing θ, since they differ only in a linear transformation.

We recommend to use $L_{min} = 1$, and tuning L_{max} in a development set. In this way, the objective in Eq.(8) with normalized loss has two appealing properties: (1) It upper bounds 0-1 loss, which makes the optimization meaningful for minimizing error rate; (2) It upper bounds the empirical risk with normalized loss $\sum_i L^s(x_i, y_i, y')$ whose minimization is equivalent to the original empirical risk $\sum_i L(x_i, y_i, y')$, which makes the optimization meaningful for minimizing the original empirical risk, the average revenue loss in our case. The key to prove both properties is $\xi_i \geq L(x_i, y_i, y')$.

Regarding implementation, the large scale of the task requires highly efficient optimization method. We solve Eq.(8) in dual space along the line of [7, 15], with a slight modification of the dual objective to reflect the desired margin as the new loss function $v(x^{(i)})L_{y^{(i)}y'}$. We choose *sequential dual* method [15] for optimization. Our implementation uses the LIBLINEAR package [12] and modifies the sequential dual solver specified with option "-s 4". Empirically, we find it very efficient on our large dataset, and is more than ten times faster than cutting-plane method [23].

An alternative to margin re-scaling is slack re-scaling (see [23] §2.2.5 for more discussion). However, our implementation with the cutting plane method in SVM^{struct} [23] is way too slow on the large dataset in our experiments. While we have very efficient sequential dual solvers in LIBLINEAR for margin re-scaling. Hence we sought to improve margin re-scaling, which is widely used in structured prediction problems [17]. Our **loss normalization** approach is applicable to general classification and structured prediction tasks, whenever using structured SVM with margin-rescaling, especially when the loss function spans a wide range.

In experiments we compare three loss normalization approaches: **IDENTITY** approach that uses original revenue in dollars, **UNIT** normalization that scales the revenue by different units, and our proposed **RANGE** normalization.

4. EXPERIMENTS

Experiments show that our cost-sensitive learning algorithm with margin re-scaling and loss normalization outperforms standard multi-class SVM, in terms of our proposed evaluation metric, average revenue loss.

4.1 Dataset

4.1.1 The UNSPSC dataset

We do experiments on a large dataset of more than 1 million products in 1073 classes. Each product has a textual description of 5 fields: *manufacturer name*; *UNSPSC code* that is the class label explained below; *product name*; *description*; and *detailed description* that is possibly empty. The dataset is collected from multiple online marketplaces oriented for Department of Defense and Federal government customers, including GSA advantage and DoD EMALL. It covers a wide range of products and services.

Each product is labeled by an 8-digit code as belonging to a leaf class in a large taxonomy. The taxonomy is called *United Nations Standard Product and Service Code* (**UNSPSC**) [1]. It is the de factor standard in US industry for hierarchical classification of general products and services. It has four levels, representing *segment*, *family*, *class*, and *commodity*, respectively. Each node in a level is specified by a 2-digit code and a text description. We identify a leaf class by an 8-digit code, concatenating the 2-digit codes along the path from the first level to the fourth level.

# examples	1,439,097
# leaf (4th level) classes	1073
# 3rd level classes	300
# 2nd level classes	99
# 1st level classes	33
avg.±std. of description length	39.5 ± 23.6
# features	784,813

Table 1: Statistics of the UNSPSC dataset

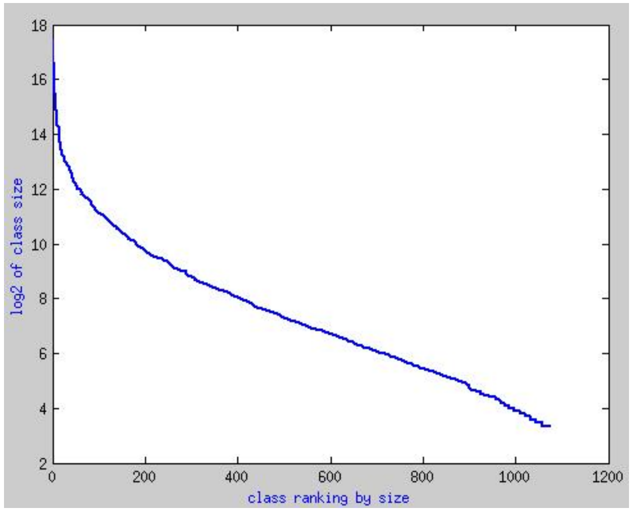

Figure 2: The class distribution in the UNSPSC dataset

The whole UNSPSC taxonomy has more than 17,000 leaf categories and is still increasing. Our dataset covers products in more than one thousand classes. We discard small classes with less than 10 products, and consider a sub taxonomy with 1073 leaf classes. The statistics of the preprocessed dataset used in our experiments is shown as Table 1. The class distribution is highly skewed as shown in Fig.Figure 2, where the X-axis is the class ranking from 1 to 1073 by size, that is, the number of examples in the class; and the Y-axis is the log2 of the class size.

4.1.2 Revenue generation

The dataset does not come with the expected annual revenue for products. So we simulate the revenue, similarly to other works in cost-sensitive learning (e.g., [9, 26]). We first generate the price and sales independently, then multiply them as the revenue.

Our price model assumes the prices of products from one class are drawn from one log-normal distribution $\ln\mathcal{N}(\mu, \sigma^2)$. Different classes have different log-normal price distributions. Log-normal distribution is used in economics to model prices [16]. To generate the parameters μ and σ^2 for each class, we use two Gamma distributions as prior distributions, respectively. Gamma distribution is commonly used as a conjugate prior for a parameter in Bayesian statistics. We choose $Gamma(k = 1, \theta = 100)$ to sample μ for each class. To generate σ^2 and control the amount of price fluctuations to be moderate for most classes, we let

$\sigma = \mu \cdot sigma\_ratio$, and prefer $sigma\_ratio$ to be more likely in $[0.2, 0.5]$. To generate such $sigma\_ratio$'s for each class, we use $Gamma(k = 30, \theta = 0.01)$. After generating μ and σ^2 for a class, we sample prices for the products in that class from the log-normal distribution $\ln \mathcal{N}(\mu, \sigma^2)$.

Similarly, our sales model assumes the sales of products from one class are drawn from one $Pareto(m, k)$ distribution. The survival function is given by $Pr(X > x) = (m/x)^k$, $m > 0, k > 0, x \geq m$, where m is the minimum possible value of x. Pareto distribution is an instance of the power law distribution that has been used to model a wide range of social and natural phenomena, including the relationship between weekly sales and sales ranks of books at Amazon.com [2]. To generate the parameters m and k for each class, we again use two priors. To sample the minimum sales m for each class, we use a Weibull distribution that is often used to model extreme value distributions. We choose $Weibull(\lambda = 2, k = 5)$ such that m has higher prior in $[1, 3]$ with thousand as unit. To sample the shape parameter k, we again use Gamma distribution. We choose $Gamma(k = 50, \theta = 0.1)$ to let k have high prior in $[4, 6]$ so that most classes have reasonably skewed sales distribution.

4.1.3 Preprocessing

Data cleaning. We remove duplicate product records in the dataset by comparing both manufacturer names and product names. We perform tokenization using simple delimiter patterns like punctuations and spaces. All tokens are transformed into lower case.

Feature extraction. We use word frequencies as features. Each token corresponds to a feature. To utilize the field information of manufacture name and product name, we add a special prefix like "$MNFT_" to each token appearing in both fields, respectively.

4.2 Results

4.2.1 Experimental setting

We randomly split the UNSPSC dataset into *training*, *development*, and *test* set by size ratios $4 : 3 : 3$, and by stratified sampling per classes. Development set is used for selecting optimal parameters. The regularization parameter C is selected from $\{0.01, 0.1, 1\}$, as we empirically find that the performance is usually best with $C = 0.1$, with monotonically decreasing performance on both sides. With larger C, the optimization also takes much longer on our large dataset. Similarly, we select revenue rescaling unit in $\{10^2, 10^4, 10^6, 10^7\}$ in dollars; loss normalization range as $[1, M_{max}]$, where M_{max} in $\{2, 5, 10, 20\}$. The best M_{max} is usually 10.

We generate 5 sets of revenues for the products using our revenue model. All results reported are averaged over 5 runs of experiments with different sets of revenue samples. Since the UNSPSC taxonomy is a 4-level balanced tree, there are only four different hierarchical distances between two leaf nodes. We specify loss ratios $L_{yy'}$ as 0.2, 0.4, 0.6, and 0.8, for hierarchical distances between the true class and the predicted class as 2, 4, 6 and 8, respectively.

4.2.2 Results and discussion

Table 2a compares the performance of multi-class SVM with margin re-scaling by *four loss functions* and *three loss normalization* approaches discussed in § 3.2, in terms of our proposed evaluation metric, average revenue loss. The table's columns correspond to different loss functions. The *0-1* loss corresponds to the *baseline*, standard multi-class SVM. The rows correspond to different loss normalization approaches.

The combination of *REVLOSS* margin re-scaling with *RANGE* loss normalization achieves the smallest average revenue loss. It reduces as much as **7.88%** average revenue loss incurred by standard SVM, which is significant with *pairwise one-tailed t-test* at significant level $p < 0.01$. Such an amount of reduction is remarkable, because it is achieved when the error rate of standard SVM is already as low as 3.8% (see Table 3), which means most products already have zero revenue loss, so any further reduction of revenue loss is non-trivial.

Comparing margin-rescaling with different *loss functions*, TREE loss increases average revenue loss, while VALUE loss achieves a significant reduction of average revenue loss, which is taken further by REVLOSS. This shows that most of the revenue loss reduction comes from exploiting the revenue of individual products; differentiating misclassification cost into different classes in REVLOSS further reduces the revenue loss slightly.

Comparing different *loss normalization* approaches (only applicable to VALUE and REVLOSS), RANGE normalization effectively improves the performance than UNIT rescaling and IDENTITY (no normalization).

To further look into what products and how the four *loss functions* with RANGE normalization tend to misclassify, Table 2b describes the mean revenue and mean tree loss (the height of the lowest common ancestor of the true class and the predicted class) of the *misclassified products* by those approaches with a single random set of revenue values. Comparing to standard SVM (0-1 loss) and TREE loss, VALUE and REVLOSS are able to tap into the product revenue information, and tend to misclassify products of significantly lower revenues on average. They tend to trade the accuracy of low revenue products for the accuracy of high revenue products; even though the final error rate of VALUE and REVLOSS is slightly higher than that of 0-1 (see Table 3), the average revenue loss of VALUE and REVLOSS is lower. On the other hand, TREE loss achieves smallest mean tree loss among the misclassified products. REVLOSS combines the advantage of both TREE and VALUE loss, yielding a desirable behavior: If it cannot classify all products correctly, it tends to sacrifice the performance of low revenue products; If it cannot classify a product into the true class, it tends to place the product into a class as close as possible.

Table 3 shows the performance of four different margin rescaling approaches with RANGE loss normalization, in terms of three common evaluation metrics. Standard SVM (0-1 loss) performs best. VALUE and REVLOSS have slightly lower but comparable performance. It might be confusing that TREE loss performs worse, even in terms of average tree loss that it is aiming to minimize. However, from 2b, TREE loss achieves smallest mean tree loss among misclassified products, so it does its job; but unfortunately it tends to increase the error rate too much, thereby increasing the average tree loss in the whole test set.

We also explored classification methods that exploits the hierarchical structure. In particular, we experimented with cascading classifiers in a top-down way. We cascade two-level classifiers. The results show similar amount of perfor-

	0-1	**TREE**	**VALUE**	**REVLOSS**
IDENTITY	4.745	4.964	47.708	48.082
UNIT			5.092	5.082
RANGE			4.387	**4.371**

(a) Average revenue loss of different algorithms

measure	**0-1**	**TREE**	**VALUE**	**REVLOSS**
mean revenue	124.4±192	116.1±185	**111.5±172**	112.9±172
mean tree loss	2.342	**2.156**	2.330	*2.328*

(b) Statistics of misclassified products by different algorithms

**All revenue loss reduction and increase in Table (a) comparing to standard SVM (0-1) are significant at* $p < 0.01$ *with pairwise one-tailed t-test at the corresponding direction (either decrease or increase loss). Revenues in both tables are of unit thousand dollar (K\$).*

Table 2: Performance of different algorithms by average revenue loss

measure	**0-1**	**TREE**	**VALUE**	**REVLOSS**
error rate	3.8	4.3	3.9	3.9
mean error rate	11.8	13.1	12.1	12.0
avg. tree loss	0.089	0.092	0.092	0.090

Table 3: Performance of different algorithms by common evaluation metrics

mance improvement with REVLOSS margin-rescaling than standard SVM, which is not shown here due to page limits. However, cascading classifiers leads to error propagation. Both error rate and mean revenue loss is slightly higher than the above flat approaches. We leave as future work to explore other hierarchical approaches like global approaches that have been shown to have higher accuracy than flat classifiers, as they leads to larger model and requires more computing resource for large taxonomies.

5. RELATED WORKS

Product classification. [19] study learning a hierarchy from the data for product classification. [14] improves product classification using images. [20] classifies product queries. Those works usually use common evaluation metrics, while we study the appropriate performance evaluation in product classification when a vendor's business goal is to maximize revenue, and the corresponding cost-sensitive learning that optimizes the proposed metric.

Hierarchical classification. On *performance evaluation* for hierarchical classification, see [6] and [21] for detailed reviews. Most works generalize evaluation metrics designed for binary classification like precision, recall, and F1-score to multi-class and hierarchical classification case. They try to be applicable to general tasks. Though we design an evaluation metric that tailors the specific task of product classification, the proposed metric has a very general form involving both example-dependent cost and class-dependent cost. Most works tend to propose metrics similar to F1 score that are non-linear function of the loss, hence they are not ideal for optimization, unlike our treatment of performance evaluation metric as empirical risk that is suitable for optimization. On *classifier learning*, see [13] for a survey of numerous works. [8, 4] extend the max-margin principle of SVM to hierarchical classification. Our experiment on cascading classifiers in a top-down way is proposed in [10].

Cost-sensitive learning. Most work studies misclassification costs that are either example-dependent [25] or class-dependent [11, 9, 26], according to Zhou's nomenclature [26]. The former give misclassification costs according to different examples. The latter give different misclassification costs according to different predicted class, while the misclassification cost into a particular false class is the same for all examples within a single class. Very few works [3] study both of them. We study misclassification cost that is both example-dependent and class-dependent. It specializes to one type of them if we set the other type of cost uniform.

6. CONCLUSION

This paper studies hierarchical product classification. In particular, we investigate two problems, *performance evaluation* and *learning*, in a synergistic way, under a unified view of empirical risk. Performance evaluation chooses an appropriate misclassification cost. Learning minimizes the average misclassification cost. We emphasizes the importance to design an appropriate performance evaluation metric for a real world task, otherwise we are optimizing the wrong objective. We show how to apply such a synergistic way to address the specific task of hierarchical product classification, and demonstrate its effectiveness by experiments on a large dataset. We obtain general insight into how and why several common evaluation metrics can be misleading, which is applicable to the treatment of performance evaluation of other real world tasks. We propose a general cost-sensitive learning algorithm that minimizes the upper bound of any loss functions, using multi-class SVM with margin re-scaling and loss normalization. The loss normalization approach is also applicable to general classification and structured prediction tasks when using structured SVM with margin re-scaling.

Our work is an application of cost-sensitive learning. Very few works study both class-dependent and example-dependent misclassification cost, especially in a practical scenario as we do. However, application scenarios involving both types of cost are not rare, even becoming more and more common in big data era, forming an emerging class of applications for large scale information and knowledge management. For example, Google Inc. detects and classifies adversarial advertisements that violates Adword policies into fine-grained classes for the benefits and safety of users [18]. The misclassification cost of an advertisement can depend on its potential revenue, and the relation between the true class and the predicted class. Another example is that a company manages numerous clients of different potential values by a taxonomy. We conjecture that such applications will become increasingly pervasive, because both taxonomies and value measures play increasingly bigger roles in modern economy and big data era. Taxonomies like Wikipedia, semantic web and patent taxonomies are widely used to organize information. On the other hand, items of monetary values abound everywhere in modern economic world. Moreover, we can assign values to them as the importance of classifying them correctly. Say in image or document classification, we assign higher values to items from certain websites to emphasize their correct classification. Discriminating values of different pieces of information takes care of important information given the sheer amount of data available nowadays.

7. REFERENCES

[1] Unspsc homepage. http://www.unspsc.org/.

[2] Chris Anderson and Mia Poletto Andersson. *The long tail*. Bonnier fakta, 2007.

[3] Alina Beygelzimer, John Langford, and Bianca Zadrozny. Machine learning techniques—reductions between prediction quality metrics. In *Performance Modeling and Engineering*, page 3–28. Springer, 2008.

[4] Lijuan Cai and Thomas Hofmann. Hierarchical document categorization with support vector machines. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, CIKM '04, page 78–87, New York, NY, USA, 2004. ACM.

[5] Yi-Wei Chen and Chih-Jen Lin. Combining SVMs with various feature selection strategies. In *Feature Extraction*, page 315–324. Springer, 2006.

[6] E. Costa, A. Lorena, ACPLF Carvalho, and A. Freitas. A review of performance evaluation measures for hierarchical classifiers. In *Evaluation Methods for Machine Learning II: papers from the AAAI-2007 Workshop*, page 1–6, 2007.

[7] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *The Journal of Machine Learning Research*, 2:265–292, 2002.

[8] Ofer Dekel, Joseph Keshet, and Yoram Singer. Large margin hierarchical classification. In *In Proceedings of the Twenty-First International Conference on Machine Learning*, page 209–216.

[9] Pedro Domingos. MetaCost: a general method for making classifiers cost-sensitive. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, page 155–164, 1999.

[10] Susan Dumais and Hao Chen. Hierarchical classification of web content. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '00, page 256–263, New York, NY, USA, 2000. ACM.

[11] Charles Elkan. The foundations of cost-sensitive learning. In *In Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, page 973–978, 2001.

[12] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: a library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008.

[13] Alex A. Freitas and Andre CPFL de Carvalho. A tutorial on hierarchical classification with applications in bioinformatics. 2007.

[14] A. Kannan, P.P. Talukdar, N. Rasiwasia, and Qifa Ke. Improving product classification using images. In *2011 IEEE 11th International Conference on Data Mining (ICDM)*, pages 310 –319, December 2011.

[15] S. Sathiya Keerthi, Sellamanickam Sundararajan, Kai-Wei Chang, Cho-Jui Hsieh, and Chih-Jen Lin. A sequential dual method for large scale multi-class linear SVMs. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, page 408–416, 2008.

[16] Cheng Few Lee, Jack C. Lee, and Alice C. Lee. Normal, lognormal distribution and option pricing model. In *Handbook of Quantitative Finance and Risk Management*, page 421–428. Springer, 2010.

[17] Ben Taskar Carlos Guestrin Daphne Roller. Max-margin markov networks. In *Advances in Neural Information Processing Systems 16: Proceedings of the 2003 Conference*, volume 16, page 25, 2004.

[18] D. Sculley, Matthew Eric Otey, Michael Pohl, Bridget Spitznagel, John Hainsworth, and Yunkai Zhou. Detecting adversarial advertisements in the wild. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, page 274–282, 2011.

[19] Dan Shen, Jean-David Ruvini, and Badrul Sarwar. Large-scale item categorization for e-commerce. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, CIKM '12, page 595–604, New York, NY, USA, 2012. ACM.

[20] Dou Shen, Ying Li, Xiao Li, and Dengyong Zhou. Product query classification. In *Proceedings of the 18th ACM conference on Information and knowledge management*, CIKM '09, page 741–750, New York, NY, USA, 2009. ACM.

[21] Aixin Sun and Ee-Peng Lim. Hierarchical text classification and evaluation. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, page 521–528, 2001.

[22] Choon Hui Teo, S. V. N. Vishwanthan, Alex J. Smola, and Quoc V. Le. Bundle methods for regularized risk minimization. *The Journal of Machine Learning Research*, 11:311–365, 2010.

[23] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, Yasemin Altun, and Yoram Singer. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6(2):1453, 2006.

[24] Vladimir Vapnik. *The nature of statistical learning theory*. springer, 1999.

[25] B. Zadrozny, J. Langford, and N. Abe. Cost-sensitive learning by cost-proportionate example weighting. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, page 435–442, 2003.

[26] Z. Zhou and X. Liu. On multi-class cost-sensitive learning. In *Proceedings of the 21st national conference on artificial intelligence*, volume 21, page 567, 2006.

[27] Cai-Nicolas Ziegler, Georg Lausen, and Lars Schmidt-Thieme. Taxonomy-driven computation of product recommendations. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, CIKM '04, page 406–415, New York, NY, USA, 2004. ACM.