Inference with the Universum

Jason Weston JASONW@NEC-LABS.COM
Ronan Collobert RONAN@COLLOBERT.COM
NEC Labs America, Princeton NJ, USA.

Fabian Sinz FABEE@TUEBINGEN.MPG.DE
NEC Labs America, Princeton NJ, USA; and
Max Planck Insitute for Biological Cybernetics, Tuebingen, Germany.

Léon Bottou LEON@BOTTOU.ORG
Vladimir Vapnik VLAD@NEC-LABS.COM
NEC Labs America, Princeton NJ, USA.

Abstract

In this paper we study a new framework introduced by Vapnik (1998) and Vapnik (2006) that is an alternative capacity concept to the large margin approach. In the particular case of binary classification, we are given a set of labeled examples, and a collection of "non-examples" that do not belong to either class of interest. This collection, called the *Universum*, allows one to encode prior knowledge by representing meaningful concepts in the same domain as the problem at hand. We describe an algorithm to leverage the Universum by maximizing the number of observed contradictions, and show experimentally that this approach delivers accuracy improvements over using labeled data alone.

1. Introduction and Motivation

In this article we study the following task: construct a function $y = f(x)$ given a set of labeled examples $\mathfrak{L} = \{(x_i, y_i)_{i=1,\dots,m}\} \in R^d \times \{\pm 1\}$ drawn from $P(x, y)$ in order to minimize the risk functional:

$$R(\alpha) = \frac{1}{2} \int |y - f(x, \alpha)| dP(x, y)$$

Now, let us suppose along with the training data we also possess a collection of unlabeled examples known not to belong to either class

$$x_1^*, \dots, x_{|\mathfrak{U}|}^*, \quad x^* \in R^d \qquad (1)$$

The set $\mathfrak{U}$ is called the *Universum*. It contains data that belongs to the *same domain* as the problem of interest and is expected to represent meaningful information related to the pattern recognition task at hand.

In the absence of a Universum, the method suggested by Statistical Learning Theory (Vapnik, 2006) is to minimize the training error whilst controlling the capacity of the set of functions $\mathcal{F}$ you use. The *Structural Risk Minimization* principle (SRM) suggests to construct a structure

$$S_1 \subset \dots \subset S_n$$

on the set of admissible functions, such that smaller indices of S are lower capacity sets of functions. One then chooses the appropriate S_k by minimizing a probabilistic upper bound on the test error of a classification model, e.g. using the VC dimension. Such a scheme justifies popular algorithms such as Support Vector Machines (SVMs) (Boser et al., 1992). SVMs perform regularization which is somehow agnostic to the data distribution, as the VC dimension is a measure of capacity that holds for all possible distributions. The structure that one constructs contains no prior knowledge of the problem.

In this article we analyse a new method for encoding prior knowledge, following Vapnik (1998) and Vapnik (2006). It works by constructing a data-dependent structure $S_1 \subset \dots \subset S_n$ on the set of admissible functions by using the Universum examples. These examples implicitly specify a prior distribution on the

Appearing in *Proceedings of the 23rd International Conference on Machine Learning*, Pittsburgh, PA, 2006.

set of functions $\mathcal{F}$, relating our approach to Bayesian approaches (Bernardo & Smith, 1994). Supplying a set of Universum examples, rather than defining such a distribution explicitly, can be a far easier task.

Universum examples should be collected to reflect information about the domain of our problem of interest. For example, if we are solving problem of digit recognition, the Universum could be objects written approximately in the same style as digits (but not necessarily digits), e.g. letters or mathematical symbols (see Figure 2). In this case, the distribution of the Universum in the feature space characterizes a domain where the dominant concepts are not pixels, but pens and strokes. Hence, the structure we construct is related to the problem at hand, and not to the arbitrary choice of a feature space.

To motivate our approach, let us first return to capacity control in the absence of a Universum, by way of the maximal margin principle.

1.1. Maximal Margin Principle

The simplest description of SLT can be obtained for the case when the set of admissible functions contains N elements $f(x, \alpha_1)), ..., f(x, \alpha_N)$. For this situation with probability $1 - \eta$ simultaneously for all N functions the following bound holds true

$$R(\alpha_i) \leq \nu(\alpha_i|\mathcal{L}) + \sqrt{\frac{\ln N - \ln \eta}{m}} \tag{2}$$

where $\nu(\alpha_i|\mathcal{L})$ is the fraction of examples in $\mathcal{L}$ incorrectly classified by function α_i.

The *Structural Risk Minimization* principle suggests to construct a structure on the set of admissible function $S_1 \subset \ldots \subset S_n$ where elements S_k contain N_k functions

$$N_1 \leq ... \leq N_n = N \tag{3}$$

One then chooses the element S_r and the function $f \in S_r$ (from the N_r possible functions) that minimizes the right hand side of equation (2).

The generalization of this scheme for an infinite number of functions is more technical than conceptual. It replaces the capacity concepts N_i with more advanced ones such as the VC dimension. Using the VC dimension capacity concept in the infinite case one obtains the same type of bounds where these concepts of capacity just replace the number of functions.

SRM is a very general scheme. The only requirement is to construct the structure *before the training data appear.* Using more sophisticated mathematics, one can obtain similar bounds without this requirement (Shawe-Taylor et al., 1998). Alternatively, using the transductive setup (Vapnik, 1998) this requirement is no longer needed.

Ignoring this technical point, the justification of the *maximum margin principle* can be achieved as follows. Suppose we choose $\mathcal{F}$ as the set of hyperplanes. Given our training set $x_1, ..., x_m$ we can factorise the infinite set of hyperplanes into a finite number

$$N = \Delta(x_1, ..., x_m)$$

of equivalence classes $\Gamma_1, ..., \Gamma_\ell$. Two functions belong to the same equivalence class if they give the same labeling on the training data.

Let us associate with each equivalence class Γ_i the margin ρ_i of the decision boundary $\mathfrak{f}_i \in \Gamma_i$ that separates the patterns $x_1, \ldots, x_n$ with the largest margin.

Therefore we obtain N pairs

$$(\mathfrak{f}_1, \rho_1), ..., (\mathfrak{f}_N, \rho_N)$$

Now let us create the following structure: we include in the element of the structure S_r all functions f for which

$$\rho_i \geq a_r, \quad a_1 > a_2, ... > a_N > 0$$

The motivation for maximal margin is that the set of hyperplanes separating data in the sphere with margin larger than a has a VC dimension less than $h < \phi(a)$. That is, by maximizing the margin we are minimizing the VC dimension, effectively the second term in equation (2).

Practically, this motivates the following algorithm, the Support Vector Machine (SVM). Support Vector Machines utilize linear discriminant functions $f_{w,b}(x) = (w \cdot x) + b$, and penalize poorly recognized patterns with the Hinge loss function (Figure 1, left), which is a convex approximation to the step function. They thus minimize:

$$\min_{w,b} \quad \frac{1}{2} \|w\|_2^2 + C \sum_{i=1}^{m} H_1[\ y_i\ f_{w,b}(x_i)\ ]$$

where $H_\theta[t] = \max\{0,\ \theta - t\ \}$ is the Hinge Loss. The regularization term $\|w\|_2^2$ causes the maximization of the margin between the two classes (Vapnik, 1998).

SVMs perform regularization which (despite being from a data-dependent class) is still somehow agnostic to the particular distribution that generates the training data, as the VC dimension is a measure of capacity that holds for all possible distributions. Indeed, the SVM regularizer $\|w\|_2^2$ bounds the norm of the gradient of the discriminant function, and hence favors "smooth" discriminant functions.

In the next section we consider another idea of assigning value to the equivalence class Γ_r using instead the Universum set. The motivation for constructing such a structure is the ability to encode prior knowledge into the capacity control mechanism of our resulting algorithm.

1.2. Maximal Contradiction on Universum Principle

Suppose that along with training data we are given another set of data, called the Universum

$$x_1^*, ..., x_{|\mathfrak{U}|}^*$$

The novelty in our procedure will now be how we construct the pairs $(\mathfrak{f}_1, \rho_1), ..., (\mathfrak{f}_N, \rho_N)$. Let us consider the set of equivalence classes Γ_r as before. We say that an element x_t^* from the Universum makes a contradiction on the equivalence class Γ_r if in Γ_r there are two functions $f(x, \alpha_1)$ and $f(x, \alpha_2)$ such that

$$f(x_t^*, \alpha_1) < 0$$

and

$$f(x_t^*, \alpha_2) > 0$$

We will count the total number of contradictions on the Universum and use this value to replace the value ρ_k from before. By defining a relevant Universum set, this will give a measure of complexity of a structure that can be related to the problem at hand, and not to the arbitrary choice of a feature space, as with the margin-based principle.

The motivation of this idea is that the number of contradictions connects the SRM principle with the use of Bayesian priors (Bernardo & Smith, 1994).

A natural way to encode prior knowledge into an algorithm is to define a prior distribution on the functions in $\mathcal{F}$. Suppose we know a prior distribution $P(w)$ on the set of hyperplanes.

We could use this to build a structure on our set of functions as follows[1]. Let us factorise our set of functions into equivalence classes as before, and define $w \in \Omega_r$ as the coefficients of hyperplanes in the equivalence class Γ_r. We can now measure the quality of an equivalence class using our prior knowledge:

$$p_r = \int_{\Omega_r} dP(w) \tag{4}$$

[1]While generalization bounds have been obtained for the large-margin structure case, for the Universum-based structure this remains an open problem. This might be possible along the lines of Shawe-Taylor et al. (1998).

The problem with this approach is that defining the distribution $P(w)$ is very hard. Using the Universum solves this problem by replacing it with an easier one: it allows the user to encode prior knowledge via a set of examples, rather than a distribution on parameters. However, defining a Universum set is approximately equivalent to choosing a distribution $P(w)$.

Let the set of hyperplanes on the space of $x \in X^r, \;\; |x| = 1$ satisfy

$$(w, x) = 0, \quad |w| = 1.$$

Taking into account the duality of x space and w space for any measure $P(w)$ there is a measure $\nu(x)$ such that the fraction of contradictive points in x approximates (4). The points in the Universum are samples from this distribution.

2. Universum Algorithm

We can now describe the algorithm for learning with a Universum.

We will use the ε-insensitive loss (Figure 1, middle):

$$U[t] = H_{-\varepsilon}[t] + H_{-\varepsilon}[-t]$$

For $\varepsilon = 0$ we have the L_1 loss. Other loss functions are possible, e.g. the L_2 loss as in Figure 1, right. This loss measures the real-valued output of our classifier $f(x)$ on Universum points $x_1^*, \ldots, x_{|\mathfrak{U}|}^*$ and penalize outputs that are far from zero. We then wish to minimize the total loss:

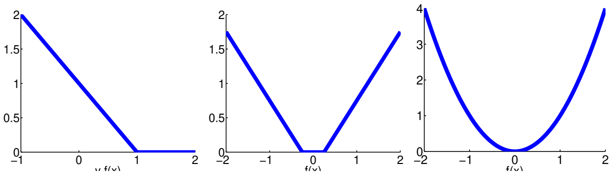
$$\sum_{i=1}^{|\mathfrak{U}|} U[f(x_i^*)]$$

This approximates our goal of finding an equivalence class with a large number of contradictions on the Universum, as if $f(x_i^*)$ is close to zero, then only a small change in f will cause a contradiction on x_i^*. There are many implementations possible, but we choose to add this term to the standard SVM objective function. That is, we minimize:

$$\frac{1}{2}\left\|w\right\|_2^2 + C\sum_{i=1}^{m} H[\,y_i f_{w,b}(x_i)\,] + C_{\mathfrak{U}}\sum_{i=1}^{|\mathfrak{U}|} U[\,f_{w,b}(x_i^*)\,]$$

We call this algorithm $\mathfrak{U}$-SVM. The loss on the Universum points enters the SVM-type optimization problem via convex constraints $|f_{w,b}(x)| \leq \varepsilon + \eta$. This optimization problem is convex, and just like SVMs the solution can also be computed in dual variables. The only difference is that the Universum loss corresponds to adding the Universum points twice with opposite label and changing the linear part of the objective function, because the Universum cost function in Figure

Figure 1. From left to right, the Hinge loss and the ε-insensitive and L_2 losses. The ε-insensitive loss is a linear combination $U[t] = H_{-\varepsilon}[t] + H_{-\varepsilon}[-t]$ of two Hinge loss functions $H_{-\varepsilon}[t] = \max\{0, t-\varepsilon\}$. Here it is shown with $\varepsilon = 0.25$. The L_2 loss is a simple quadratic function.



1 (middle) is a symmetrized version of the hinge loss, Figure 1 (left).

For $i = 1 \ldots |\mathfrak{U}|$, let us define:

$$\begin{aligned} x_{m+i} &= x_i^* & y_{m+i} &= +1 \\ x_{m+|\mathfrak{U}|+i} &= x_i^* & y_{m+|\mathfrak{U}|+i} &= -1 \end{aligned}$$

After some algebra, the problem becomes:

$$\max_{\alpha} \quad \sum_{i=1}^{m+2|\mathfrak{U}|} \rho_i \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m+2|\mathfrak{U}|} y_i y_j \alpha_i \alpha_j \, (x_i \cdot x_j)$$

$$\text{s.t.} \begin{cases} 0 \le \alpha_i \le C & \text{for } i = 1 \ldots m \\ \rho_i = 1 & \text{for } i = 1 \ldots m \\ 0 \le \alpha_i \le C_{\mathfrak{U}} & \text{for } i = m+1 \ldots m+2|\mathfrak{U}| \\ \rho_i = -\varepsilon & \text{for } i = m+1 \ldots m+2|\mathfrak{U}| \\ \text{and} \quad \sum_{i=1}^{m+2|\mathfrak{U}|} y_i \, \alpha_i \; = \; 0 & \end{cases}$$

We note that a similar optimization problem is considered in Zhong and Fukushima (2006), but with quite different motivations.

Collecting Universum examples We believe it should be easy to collect or construct Universum data for many different types of problems. We already gave the example of optical character recognition. Some other examples include 3D object detection (any set of objects could be used to learn about stereo vision), speech recognition (languages other than the one of interest could be Universum examples) and so on.

In cases where Universum data is not easily available in abundance, one can instead use a priori domain knowledge to construct purely artificial examples. One could potentially construct fake handwritten symbols by simulating pens and strokes, fake 3D object by simulating the stereo mapping, or synthesize fake sounds. We explore both real and synthesized Universum examples in our experiments.

3. Regularization with the Universum

The term $\sum_{i=1}^{|\mathfrak{U}|} U[\, f(x_i^*) \,]$ in $\mathfrak{U}$-SVM that takes into account the Universum points can be seen as a regularizer defined by the Universum data.

This section explores how we can recover a wide range of known regularizers by defining special sets of Universum points. We also describe some novel regularization strategies.

Isotropic L_2 regularization. Let us consider a linear classifier without threshold, $f_w(x) = w \cdot x$, and apply the L_2 Universum loss $U[f_{w,b}(x_i^*)] = |f_{w,b}(x_i^*)|^2$ (Figure 1, right) to n Universum examples x_k^* whose coefficients are all zero apart for the k^{th} which is 1.

$$\sum_{i=1}^{|\mathfrak{U}|} U[\, f_w(x_i^*) \,] = \sum_{i=1}^{|\mathfrak{U}|} (w \cdot x_i^*)^2 = \sum_{k=1}^{n} w_k^2 = \|w\|_2^2$$

One recognizes the standard L_2 regularizer.

Anisotropic L_2 regularization. More generally, take a linear classifier $f_{w,b}(x) = w \cdot x + b$ and apply the L_2 Universum loss to Universum examples with mean 0 and covariance matrix M.

$$\begin{aligned} \sum_{i=1}^{|\mathfrak{U}|} U[\, f_{w,b}(x_i^*) \,] &= \sum_{i=1}^{|\mathfrak{U}|} (\, w^\top x_i^* \;+\; b \,)^2 \\ &= w^\top \left(\sum_{i=1}^{|\mathfrak{U}|} x_i^* x_i^{*\top} \right) w + 2b \, w^\top \left(\sum_{i=1}^{|\mathfrak{U}|} x_i^* \right) + \left(\sum_{i=1}^{|\mathfrak{U}|} b^2 \right) \\ &= |\mathfrak{U}| \; (\, w^\top M \, w \;+\; b^2 \,) \end{aligned}$$

This regularizer uses the L_2 metric weighted by the covariance matrix M of the Universum examples. When the covariance matrix is diagonal, this amounts to whitening this covariance matrix by rescaling the features. This is remininiscent of the TF/IDF normalization in text processing where common features are downweighted as rare features prove more useful for discrimination.

This regularizer also penalises the threshold b. Intuitively, the center of mass of the Universum points must be located at a specific position in feature space. The Universum cannot implement a translation invariant regularizer. This is connected to the fact that one cannot define a uniform distribution on the whole affine space. This relates to the use of "improper priors" in Bayesian setups.

L_1 regularization - linear case It is also possible to implement the L_1 regularization that is commonly used for feature selection (Mangasarian, 1965). Consider a linear classifier without threshold $f_w(x)$ and apply the L_1 Universum loss, $U[t] = H_0[t] + H_0[-t]$, (Figure 1 center) to n Universum points x_k^* whose coefficients are all zero apart from the k^{th} which is 1.

$$\sum_{i=1}^{|\mathfrak{U}|} U[\, f_w(x_i^*)\,] = \sum_{i=1}^{|\mathfrak{U}|} |w \cdot x_i^*| = \sum_{k=1}^{n} |w_k| = \|w\|_1$$

One recognizes the standard L_1 regularizer.

L_1 regularization – kernel case Usually one cannot implement the L_1 regularizer with nonlinear kernel classifiers because the dual formulation does not apply and because the high dimension of the kernel induced feature space makes the primal formulation too costly to compute.

Nevertheless the Universum formulation suggests a practical way to implement a form of input selection in the nonlinear case: simply take the Universum described in the section above. This will still perform input selection even for nonlinear kernels. Consider for instance a polynomial kernel of degree d defined on binary input variables. In this case this corresponds to an L_1 regularizer that applies only to the coefficients corresponding to the linear part of the decision function.

4. Experimental Analysis

In this section we test experimentally whether inference using Universum points is beneficial compared to standard supervised learning. In all cases, we compare a standard SVM to $\mathfrak{U}$-SVM, that is, to an SVM that also leverages Universum data. Some of our experiments also try to explore the question: what kind of Universum is useful, and when? Unless described otherwise, we employ an RBF kernel, with the width and soft-margin hyperparameter C tuned using a validation set. For $\mathfrak{U}$-SVM, we also tune the regularization parameters ε and $C_{\mathfrak{U}}$. online before the conference.

4.1. MNIST

We first took the MNIST digits 5 vs 8 as a two-class classification problem, to see the performance of $\mathfrak{U}$-SVM on a standard dataset. For this problem we considered four kinds of Universum:

(i) $\mathfrak{U}_{Noise}$ - images of "random noise" by generating uniformly distributed pixel features,

(ii) $\mathfrak{U}_{Rest}$ - the other digits 0-9 excluding 5 and 8,

(iii) $\mathfrak{U}_{Gen}$ - create an artificial image by generating each pixel according to its discrete empirical distribution on the training set.

(iv) $\mathfrak{U}_{Mean}$ - create an artificial image by first selecting a random 5 and a random 8 from the training set, and then constructing the mean of these two digits.

Table 1. The performance of SVM compared to $\mathfrak{U}$-SVM for different amount of training data and different types of Universum data on MNIST 5 vs 8.

	Training subset size			
Method	500	1000	2000	3000
SVM	1.96	1.38	0.99	0.83
$\mathfrak{U}_{Noise}$-SVM	1.95	1.37	0.99	0.82
$\mathfrak{U}_{Rest}$-SVM	1.60	1.10	0.75	0.55
$\mathfrak{U}_{Gen}$-SVM	1.72	1.17	0.81	0.64
$\mathfrak{U}_{Mean}$-SVM	1.68	0.99	0.73	0.57

Table 2. The performance of $\mathfrak{U}_{Mean}$-SVM for differing amounts of Universum data on MNIST 5 vs 8. SVM performance for the same number of training points (3000) was 0.83%.

Train.	Number of Universum examples				
examples	500	1000	3000	5000	10000
3000	0.66	0.64	0.60	0.57	0.58

$\mathfrak{U}_{Noise}$ was included as a kind of "null" hypothesis to show that not just any Universum helps – it has to be related to the problem of interest. The results for different training set subset sizes are reported in Figure 1[2]. They show an improvement of $\mathfrak{U}$-SVM over SVM for every Universum apart from $\mathfrak{U}_{Noise}$.

Which part of the Universum is useful? Next, we tried to ascertain which digits from the Universum $\mathfrak{U}_{Rest}$ were the most useful in improving the classification accuracy. The initial intuition is that the digits that are close to 5 and 8 should help most. We report the best test error on a test set of 1865 digits for algorithms trained on the whole training set of 11271 digits and averaged over ten training sets of size 1000 and 200 that we sampled from the original training set. We always used the whole set of digits from one class as the Universum. (This set up is slightly different from before.) The sizes of these sets are around 6000 examples for each digit. The results are given in Table 3. They indicate that digits "3" and "6" are the most useful. This seems to match our intuition, as these digits seem somehow "in between" the digits "5" and "8", whereas

[2] In Vapnik (2006) a similar experiment is reported, but with differently constructed Universum data.

Table 3. Test error rates for MNIST 5 vs. 8 using examples of one other digit (0,1,2,3,4,6,7 or 9) as the Universum set. The last two columns show the correlation of Universum points with examples of 5s and 8s.

$\mathfrak{U}$	Training subset size			Correlation	
	all	1000	200	ρ_5	ρ_8
0	0.27	0.97	3.03	0.32	0.29
1	0.16	1.01	2.95	0.24	0.36
2	0.21	0.94	3.21	0.24	0.34
3	0.05	0.62	2.97	0.33	0.37
4	0.21	0.93	3.03	0.27	0.32
6	0.16	0.84	2.40	0.26	0.32
7	0.16	1.08	3.23	0.25	0.30
9	0.21	0.89	2.78	0.30	0.37
-	0.21	1.19	3.03	-	-

other digits have this property only to a lower degree (mean correlation coefficients with examples of digits 5 and 8 are given in Table 3). Roughly speaking one can say that the Universum loss penalizes features that have high values on the Universum points. The digit "3" covers most of the parts that appear in the digit "5" as well as in "8" which can therefore be considered less discriminative. Taking class 3 as Universum is a good choice for improving accuracy by assigning less relevance to those less discriminant features than any of the other classes. Beyond that, the digits in a class are not perfectly aligned but rather subject to transformations like rotation or translation. Ideally a classifier should be invariant against those. But since the Universum points are subject to those transformations as well, the features that are affected are also assigned lower importance. This could also explain why an SVM using other digits as Universum also improves the performance over that of a plain SVM in most cases.

4.2. Reuters RCV1-V2

The Reuters dataset consists of over 800,000 news articels in English languages written by Reuters journalists between August 20, 1996 and August 19, 1997. We used the freely accessible preprocessed version of Lewis et al. (2004).

The task was to separate the category C15 from the remaining categories at the same level of the hierarchy, category CCAT (CORPORATE/INDUSTRIAL). The data was represented as a bag of words weighted by a TF/IDF scheme and normalized to Euclidean length one (see Lewis et al. (2004) for details).

We split the set of 13310 examples into a training set of 6000 examples, a validation set of 2000 examples and a test set of 5310 examples to accelerate model selection. We generated subsets of sizes 50, 100, 200, 500 and 1000. Ten sets for each size were randomly selected from the 6000 points.

Table 4. Test error in percent on Reuters RCV1 for SVMs and $\mathfrak{U}$-SVMs, with M14- and MoC-Universums, for different training set sizes.

	Training subset size				
Method	50	100	200	500	1000
SVM	21.1	13.1	11.0	8.6	7.6
$\mathfrak{U}_{M14}$-SVM	15.7	12.7	10.2	8.2	7.6
$\mathfrak{U}_{MoC}$-SVM	19.4	12.6	10.8	8.6	7.6

Table 5. Test errors in percent comparing SVM and $\mathfrak{U}_{Mean}$-SVM on the WinMac dataset.

	Training subset size				
Method	10	25	50	75	100
SVM	45.2	31.7	20.3	14.7	11.7
$\mathfrak{U}_{Mean}$-SVM	33.0	24.3	15.2	12.3	11.0

We chose two kind of Universums, a real and an artificial one. We chose the category M14 (COMMODITY MARKETS) with 2540 examples as the real Universum. For the artificial Universum, we selected $N = 10$ examples from each class of the training set and added the mean of the closest examples from to different classes to the Universum. Altogether, we generated 1000 points in this manner. In the following text we call that Universum the MoC Universum (mean of closest). It might be worth noting that we generated a MoC Universum for each single split in order not to use additional information from other training examples.

We used an RBF kernel, since preliminary tests showed that it perfoms slightly better than the linear kernel. The tuning of the regularization constants $C, C_{\mathfrak{U}}$ and the kernel parameter was done on the first of the ten sets with a model selection using the validation set. For the best set of parameters we trained an SVM with and without Universum on each of the ten subsets for the different training set sizes and tested each on the test set.

Table 4 shows the averaged results. Both $\mathfrak{U}$-SVMs perform better than a plain SVM. For a dataset size of 50 the improvement is in the order of 1% for the MoC Universum and in the order of 5% for the M14 Universum. With increasing dataset size the effect of the Universum vanishes. These results suggest that the prior knowledge from the M14 Universum really is important for the classifier. Especially for small dataset sizes, the M14 Universum can exhibit features that are not discriminative for the classification problem since they seem to occur throughout the dataset. As soon as the dataset has an adequate size, it provides enough information itself and the effect of the Universum disappears.

Similar results can be found on other text datasets as well, such as the WinMac dataset, a collection of newsgroup articles in two categories. Here, we took 10 random splits of training subset sizes 10, 25, 50, 75, 100, using a linear kernel. The results are given in Table 5. For larger training set sizes, the improvement again becomes negligible.

4.3. The ABCDETC Dataset

Most standard machine learning datasets of course do not come equipped with Universum data. For example, MNIST only contains digits of interest, forcing us to run somewhat artificial experiments on that dataset. Despite this, Universum data is in fact quite easy to collect. We therefore decided to collect our own handwritten symbol-based dataset comprising of digits, upper and lower case letters, and a selection of symbols:

, . ! ? ; : = − + / ( ) $ % " @

Thus we collected 78 classes in all. Subjects wrote in pen 5 versions of each symbol on a single gridded sheet. The sheets were scanned at 300dpi, and the symbols were stored as 100 x 100 patches, which were automatically extracted and then centered using the center of mass of the pixels. In the following experiments, there are 51 subjects resulting in a dataset of 19,646 examples, after outlier removal. Figure 2 shows part of a typical sheet entered by a subject. The current dataset, and updates as we plan to expand it, will be available online before the ICML conference.

We performed experiments on predicting whether a letter is a lower case "a" or "b", using training sets of various sizes (20, 50, 100, 150 and 200), a validation set of size 200, and the remaining data as the test set (between 100-300 examples, depending on the size of the training set). We report results averaged over 10 random splits. We normalized the examples to have length 1, and chose to use polynomial kernels, $K(x, y) = (x \cdot y + 1)^d$. We compare standard SVMs to $\mathfrak{U}$-SVMs using four different Universum sets: (i) the set of remaining lower case letters, (ii) the set of upper case letters C-Z, (iii) the set of digits; and (iii) the set of symbols. In all cases we randomly sampled 1500 points so all the Universum sets are the same size. The

Figure 2. Part of a typical scanned sheet used to compile the ABCDETC dataset. The fourth row tells the subject what to write in the five rows below it.

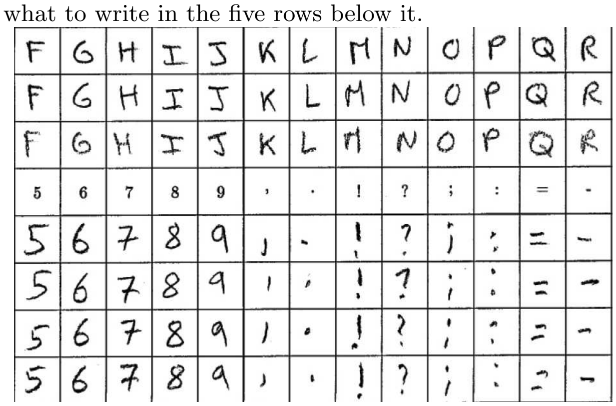


Table 6. Comparison of SVM and $\mathfrak{U}$-SVM learning lower case "a" versus "b" on the ABCDETC dataset, a collection of handwritten letters, digits and symbols. Four different Universum choices are considered: lower case c-z, upper case C-Z, digits 0-9, or a selection of symbols.

	Training subset size				
Method	20	50	100	150	200
SVM	9.93	5.71	5.16	4.53	3.85
$\mathfrak{U}_{Lowcase}$-SVM	8.75	5.09	4.21	3.89	3.39
$\mathfrak{U}_{Upcase}$-SVM	8.79	5.52	4.88	3.65	2.84
$\mathfrak{U}_{Digits}$-SVM	8.37	5.56	4.26	3.97	3.49
$\mathfrak{U}_{Symbols}$-SVM	8.62	5.75	5.17	4.40	3.67

results are reported in Table 6. They show that as the Universum set becomes intuitively "less relevant" to the problem at hand, the gain one gets from using it decreases.

4.4. Feature Selection Toy Datasets

We next tested the feature selection regularization of taking a Universum set $x_i^* = (0, \ldots, 0, 1, 0, \ldots, 0)$, where there is a 1 in the i^{th} dimension. In the linear case this is equivalent to adding a 1-norm regularizer, in the nonlinear case it also penalizes using many input features. We constructed two toy problems: a linear one with 2 relevant features in an AND problem and 18 noise inputs, and a nonlinear one with 2 relevant features in an XOR problem with 4 noise inputs. All input features are generated from a uniform distribution. We generated 50 training points, a validation set of size 200, and a test set of size 1000, for 10 separate splits. We report error rates for linear, polynomial and RBF kernels for both SVMs and $\mathfrak{U}$-SVMs, where we tuned kernel hyperparameters, C and $C_{\mathfrak{U}}$ on the validation set.

Table 7. Comparison of SVM and $\mathfrak{U}$-SVM using an L_1-based Universum set on two toy feature selection problems.

	Toy problem	
Method	Linear	Non-Linear
SVM_{linear}	16.0	49.2
SVM_{poly}	15.6	23.0
SVM_{rbf}	14.4	23.8
$\mathfrak{U}_{L_1}\text{-SVM}_{linear}$	6.2	48.5
$\mathfrak{U}_{L_1}\text{-SVM}_{poly}$	6.2	12.1
$\mathfrak{U}_{L_1}\text{-SVM}_{rbf}$	6.3	19.2

The results given in Table 7 show a considerable improvement of $\mathfrak{U}$-SVMs over SVMs. However, we note that many other feature selection algorithms exist. We do not claim that this is the best one, but show it as another illustrative example of how constructing a Universum set can realize many different types of regularization.

5. Discussion

The idea of adding new data to an existing training set in order to get better performance was used in several different settings of the pattern recognition problem. In transductive and semi-supervised learning one leverages unlabeled data from the same distribution. In the virtual examples methods (Baird, 1990; Leen, 1995; Schölkopf et al., 1996; Niyogi et al., 1998) and noise injection (Grandvalet et al., 1997), on the other hand, one creates labeled synthetic data that may not come from the same distribution.

The idea of using a Universum is also about the use of additional data. However here we do not require either the same distribution or labelling.

The Universum idea is close to the Bayesian idea: the attempt to use prior knowledge. However there is a conceptual difference between the two approaches. In Bayesian inference the prior knowledge is knowledge about decision rules, while the Universum is knowledge about the admissible collection of examples. People may have some feeling about a set of examples but they may often know nothing about the distribution on the admissible set of functions. Like the Bayesian prior, the Universum encodes prior information. Unlike the Bayesian prior, the Universum distribution does not depend on the admissible family of functions.

Our experiments show that the obtained performance depends on the quality of the Universum. The methodology of choosing the appropriate Universum is the subject of research. However our results confirm that the Universum can be an important instrument for boosting performance, especially in the small sample size regime.

Acknowledgements We thank Olivier Chapelle and Bernhard Schölkopf for helpful discussions and support. Part of this work was funded by NSF grant CCR-0325463.

References

Baird, H. (1990). Document image defect models. *Proceedings, IAPR Workshop on Syntactic and Structural Pattern Recognition* (pp. 38–46). Murray Hill, NJ.

Bernardo, J. M., & Smith, A. F. M. (1994). *Bayesian theory.* John Wiley and Sons.

Boser, B. E., Guyon, I. M., & Vapnik, V. (1992). A training algorithm for optimal margin classifiers. *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory* (pp. 144–152). Pittsburgh, PA: ACM Press.

Grandvalet, Y., Canu, S., & Boucheron, S. (1997). Noise injection: Theoretical prospects. *Neural Computation*, *9*, 1093–1108.

Leen, T. K. (1995). From data distributions to regularization in invariant learning. *Advances in Neural information processing systems 7.* Cambridge MA: MIT Press.

Lewis, D. D., Yang, Y., Rose, T., & Li, F. (2004). Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, *5*, 361–397.

Mangasarian, O. L. (1965). Linear and nonlinear separation of patterns by linear programming. *Operations Research*, *13*, 444–452.

Niyogi, P., Girosi, F., & Poggio, T. (1998). Incorporating prior information in machine learning by creating virtual examples. *Proceedings of the IEEE*, *86*, 2196–2209.

Schölkopf, B., Burges, C., & Vapnik, V. (1996). Incorporating invariances in support vector learning machines. *Artificial Neural Networks — ICANN'96* (pp. 47–52). Berlin: Springer Lecture Notes in Computer Science, Vol. 1112.

Shawe-Taylor, J., Bartlett, P. L., Williamson, R. C., & Anthony, M. (1998). Structural risk minimization over data-dependent hierarchies. *44*, 1926–1940.

Vapnik, V. (2006). *Estimation of dependences based on empirical data.* Berlin: Springer Verlag. 2nd edition.

Vapnik, V. N. (1998). *Statistical learning theory.* New York: Wiley.

Zhong, P., & Fukushima, M. (2006). A new multi-class support vector algorithm. *Optimization Methods and Software*, *21*, 359–372.