Training a Support Vector Classifier using a Cauchy-Laplace Product Kernel

P. Chandrasekhar, Ph.D
Dept of CSE, S.K University
Ananthapur, A.P, India

B. Mallikarjuna Reddy
Assistant Professor
Dept of CSE, PVKK Institute of Technology
Ananthapur, A.P, India

ABSTRACT
The importance of the support vector machine and its applicability to a wide range of problems is well known. The strength of the support vector machine lies in its kernel. In our recent paper, we have shown how the Laplacian kernel overcomes some of the drawbacks of the Gaussian kernel. However this was not a total remedy for the shortcomings of the Gaussian kernel. In this paper, we design a Cauchy-Laplace product kernel to further improve the performance of the Laplacian kernel. The new kernel alleviates the deficiencies more effectively. During the experimentation with three data sets, it is found that the product kernel not only enhances the performance of the support vector machine in terms of classification accuracy but it results in obtaining higher classification accuracy for smaller values of the kernel parameter γ. Therefore the support vector machine gives smoother decision boundary and the results obtained by the product kernel are more reliable as it overcomes the problems of over fitting.

Keywords
Support Vector Machine, hyper plane, Mercer kernel, Laplacian kernel, Cauchy-Laplace product kernel.

1. INTRODUCTION
Classification of tremendous amount of data is time consuming and utilizes excessive computational effort, which may not be appropriate for many applications. The high number of features in the image space, a large number of training samples required and the Huges phenomena are all limitations over classifiers. Since the beginning of 21st century, classifiers based on statistical learning theory have shown remarkable abilities to deal with both high-dimensional data and a limited training set. One of the best-known methods is the support vector machines (SVMs) which has shown promising empirical performance. This is because a number of attractive features are overly associated with them: ideas of support and non-support training vectors, weighting training data, discounting data, regularization, margin and bounding of the generalization error. These are all important enough ideas to stand on their own and are often seen in simpler settings. The formulation embodies the Structural Risk Minimization (SRM) principle, as opposed to the Empirical Risk Minimization (ERM) approach commonly employed within statistical learning methods. SRM minimizes an upper bound on the generalization error, as opposed to ERM which minimizes the error on the training data. It is this difference which equips SVMs with a greater potential to generalize. The SVM can be applied to both classification and regression problems.

What makes SVM attractive is the property of condensing information in the training data and providing a sparse representation by using a very small number of data points (SVs) (Girosi, 1998). SVM is a linear classifier in the parameter space, but it is easily extended to a nonlinear classifier of the φ-machine type (Aizerman, Braverman & Rozonoer, 1964) by mapping the space of the input data X= {x} into a high-dimensional (possibly infinite-dimensional) feature space F = {φ(x)}. By choosing an adequate mapping φ, the data points become linearly separable or mostly linearly separable in the high-dimensional space, so that one can easily apply the structure risk minimization. We need not compute the mapped patterns φ(x) explicitly, and instead we only need the dot products between mapped patterns. They are directly available from the kernel function which generates φ(x). The performance of SVM largely depends on the kernel. Smola, Scho¨lkopf and Mu¨ller (1998) elucidated the relation between the SVM kernel method and the standard regularization theory (Girosi, Jones & Poggio, 1995). The strength of a support vector machines lies in its kernel. However, there are no theories concerning how to choose good kernel functions in a data dependent way.

2. THE LAPLACIAN KERNAL
Before describing our problem, it is necessary to examine some properties of the Laplacian kernel. The properties are similar to the Gaussian RBF kernel which is preferentially used as the SVM kernel. The output of the kernel is dependent on the Euclidean distance of x from x' (one of these will be the support vector and the other will be the testing data point). The support vector will be the centre of the Laplacian RBF and γ will determine the area of influence this support vector has over the input data space. The Laplacian kernel $k(x, x') = \exp(-\gamma \|x - x'\|)$ is essentially zero if the distance $\|x - x'\|$ is much larger than $1/\gamma$. That is, for a fixedx', vectors in a narrow region around a support vector alone are important because outside this region $\exp(-\gamma \|x - x'\|)$ is zero. Thus the kernel is localized to smaller regions aroundx'. Consequently the discriminant function f(x) (which defines a hyperplane and is a summation involving kernel functions) is localized to smaller regions around x' resulting in greater curvature of the decision boundary. In this regime of the γ parameter the classifier clearly over fits the data. This results in bumps centered around each support vector.

When γ is large the value of the discriminant function is essentially constant outside the close proximity of the region where the data are concentrated.

When γ is small or comparable to 1, a given data point x has a non-zero kernel value relative to any example x' in the set of support vectors. Therefore the whole set of support vectors will affect the value of the discriminant function at x, resulting in a smooth decision boundary. As γ increases, as

described in the above paragraph, the locality of the support vector expansion decreases, leading to greater curvature of the decision boundary. In general, the function that imbeds the original space into the augmented feature space is unknown. The existence of such function is however assured by Mercer's theorem (Mercer 1909). The effect of such function is confined within the constructed kernel, which must express a dot product of the function φ from the input space to the feature space. Moreover, all used kernels in the literature are either dot product functions or distance functions. By adopting the latter formulation, knowing an estimation of the Euclidean distance between two points in the original space, we examine how much they are correlated in the augmented space. In most commonly distance based kernels, points very close to each other are strongly correlated particularly when the distance appears in the exponential. Our concern is to force the images of the original points to be linearly separable in the augmented space. In order to get such a behavior, a kernel $k(x,x') = \varphi(x).\varphi(x')$ must be so as to force the function φ to turn very close points from the original space into weakly correlated elements (as weak as possible) in the image space, while still maintaining the closeness information from vanishing. To achieve this tradeoff, we need the following couple of features: a quick decrease in the neighborhood of zero and a moderate decrease toward infinity. The Laplacian kernel is comparatively slow in descending from 1 as the distance increases from zero. This can result in information loss in the implementation of classification, particularly when the data points are too wide or too narrow in the sense of the distance.

3. MODEL SELECTION

We make the following two observations:

1. For all points x such that $\|\mathbf{x}-\mathbf{x}'\|$ is small ($\ll 1$), the exponential kernel function is nearly constant (almost zero) and results in tight relationship (or correlation) among such points. This results in information loss inside the SVM model through the kernel application. Separation of points in the feature space suffers a constraint. Relations should be absent in the image space to achieve successful separation. We have to modify the kernel so as to overcome this difficulty. Therefore, over a small domain $0 < \|\mathbf{x}-\mathbf{x}'\| < \epsilon$, the kernel function should quickly decrease from 1 as $\|\mathbf{x}-\mathbf{x}'\|$ increase from 0 to ϵ, compared to the exponential kernel.

2. For all points x such that $\|x-x'\| \gg 1$, i.e., for points which are far separated, the exponential kernel $k(x,x') = \varphi(x).\varphi(x')$ is very nearly zero. This kind of correlation is also a limitation for free dispersal of points $\varphi(x)$ in the feature space. The property of the kernel to quickly decrease towards zero and to be zero nearly zero for all $\|x-x'\|$ beyond a stage should be overcome by the new kernel.

In fact, the first condition is more prone to occur in data sets because consistent data items normally do not differ by large values.

Therefore, we need the following couple of features for the new kernel function: a quick decrease in the neighborhood of zero and a moderate decrease toward infinity (slower) when compared to the exponential kernel. We have to modify the exponential kernel to address these issues.

Motivated by the above aspects, we propose to design a kernel which overcomes at least one of these shortcomings and will be able to enforce better classification for moderately large data sets. We propose the following kernel and describe how it overcomes these drawbacks. We consider the product kernel given by $\frac{1}{1+\|x-x'\|}\exp(-\gamma\|x-x'\|)$. The kernel is the product of two kernels, namely the Cauchy kernel. $\frac{1}{1+\|x-x'\|}$ (Sangeetha and Kalpana 2011) and the Laplacian kernel$\exp(-\gamma\|x-x'\|)$. Both these functions are distance based kernel functions and monotonic decreasing functions of the distance $r = \|x-x'\|$ on the interval $[0,\infty)$ whose values lie in the interval [0,1]. Both these functions take the value 1 when $r = 0$ and 0 when$r \to \infty$. Thus the product kernel also decreases from 1 to 0 as r increases. The above kernel is a valid Mercer kernel being the product of two Mercer kernels. All the diagonal entries in the kernel matrix are 1 and all other entries are less than the diagonal entries.

We check whether the merit of two kernels combined to form a hybrid kernel can result in an improvement. It is so expected because the Cauchy kernel is a long tailed kernel and can be used to give long-range influence and sensitivity over the high dimension space (Basak 2008). For a given $\|x-x'\|$, each of the two functions is less than 1 and their product is much less than the value returned by the Laplacian. Therefore the condition (1) above will be surely met. It reminds us that the product of two kernels is like the Boolean AND operation. Higher classification accuracy can be achieved by optimizing the new kernel function and tuning its kernel parameters. The experimentation has shown that, in comparison with the Laplacian, the kernel is able to achieve better classification. Thus the product kernel function has resulted to overcome the drawbacks of undue sparseness and robustness (unduly sparse or dense), resulting in wider and uncorrelated disbursement of image points in the feature space. Theoretically also we can derive the strength of the product kernel. Let k_1 be the Laplacian and k_2 be the product kernel. Then if we let $\|x-x'\| = r$, and make expansions of these functions for $r \ll 1$, we find that the Laplacian kernel behaves as $1-\gamma r$ whereas the product behaves as $1-(1+\gamma)r$ and we know that $1-(1+\gamma)r$ is smaller than $(1-\gamma r)$ when $\gamma \sim O(1)$. This means the product kernel curve decreases quickly as r increases in a narrow region $0 < r < \epsilon$. In fact this prompted us to construct the modified product kernel to remedy the situation arising in the Laplacian kernel when r is small.

However, the product kernel cannot do much to prevent the sparseness requirement, i.e., behavior for large $\|x-x'\|$ given by the condition (2). All the points x which are far separated in the input space from a test point x' have very nearly zero kernel value. Our kernel is not able to do much in this regard. Therefore we expect our kernel to serve better purpose (relative to the Gaussian) only for data sets in which $\|x-x'\|$ is not very large; i.e., examples are not too much separated in the input space. However, it cannot be worse than training SVM with the Gaussian or a Laplacian. Motivated by these ideas, we attempt the problem of classification of data sets using a Support Vector Machine using the product kernel. The data sets considered are described during the discussion. Now we present a very brief overview of linear SVM followed by non-linear SVM.

4. SVM CLASSIFIER

Support vector machines are an example of a linear two-class classifier. The data for two class learning problem consists of objects labeled with one of two labels corresponnding to the two classes; for convenience we assume the labels are +1 (positive examples) or -1 (negative examples). In what follows boldface x denotes a vector with components

x_i. The notation x_i will denote the i th vector in a data set $\{(x_i, y_i); i = 1,2, \dots, n\}$ where y_i is the label associated with x_i The objects x_i are called patterns or examples. We assume that the examples belong to some set X. Initially we assume that the examples are vectors, but once we introduce kernels, this assumption will be relaxed at which point they could be any continuous or discrete objects.

A key concept required for defining a linear classifier is the dot product between two vectors, also referred to as an inner product or scalar product defined as $w^T x = \sum_i^n w_i x_i$. A linear classifier is based on a linear discriminant function of the form$f(x) = w^T x + b$

The vector w is known as the weight vector, and b is called the bias. Consider the case $b = 0$ first. The set of points x such that $w^T x = 0$ are all points that are perpendicular to w and go through the origin – a line in two dimensions, a plane in three dimensions, and more generally a hyperplane. The bias b translates the hyperplane away from the origin. The hyperplane $\{x : f(x) = w^T x + b\} = 0$

divides the space into two.The sign of the discriminant functio n f(x) in Eq.(2) denotes the side of the hyperplane a point is on. The boundary between regions classified as positive and negative is called the decision boundary of the classifier. The decision boundary defined by a hyperplane is said to be linear because it is linear in the input. A classifier with a linear decision boundary is called a linear classifier. Conversely, when the decisioin boundary of a classifier depends on the data in a non-linear way the classifier is said to be non linear.

5. NON LINEAR CLASSIFIER

The machinery of linear classifiers can be extended to generate non-linear decision boundaries. The naïve way of making a non-linear classifier out of linear classifier is to map our data from the input space X to a feature space F using a non-linear function $\varphi: X \to F$. In the feature space F the discriminant function is $f(\mathbf{x}) = \mathbf{w}^T \varphi(\mathbf{x}) + b$ (3) Kernel methods avoid the step of explicitly mapping the data to a higher dimensional feature space. Suppose the weight vector can be expressed as a linear combination of the training examples, that is, $\mathbf{w} = \sum_{i=1}^n \alpha_i \mathbf{x}_i$. Then, $f(\mathbf{x}) = \sum_{i=1}^n \alpha_i \mathbf{x}_i^T \mathbf{x} + b$. In the feature space F, this expression takes the form $f(\mathbf{x}) = \sum_{i=1}^n \alpha_i \varphi(\mathbf{x}_i)^T \varphi(\mathbf{x}) + b$. The representation in terms of the variables α_i is known as the dual representation of the decision boundary. As indicated above, the feature space F may be high dimensional, making this trick impractical unless the kernel function $k(\mathbf{x}, \mathbf{x}')$ is defined as $k(\mathbf{x}, \mathbf{x}') = \varphi(\mathbf{x})^T \varphi(\mathbf{x}')$ - a dot product – that can be computed efficiently. In terms of the kernel function, the discriminant function is $f(\mathbf{x}) = \sum_{i=1}^n \alpha_i k(\mathbf{x}, \mathbf{x}') + b$.

We saw that a linear decision boundary can be "kernalized", i.e., its dependence on the data is only through dot product. In order for this to be useful, the training algorithms need to be kernalized as well. It turns out that a large number of machine learning algorithms can be expressed using kernels – including ridge regression, the perceptron algorithm and SVMs (Scho'lkopf and Smola (2002) and Cristianini and Shawe-Taylor (2000)).

We use the term linearly seperable to denote data for which there exists a linear decision bounday that seperates positive from negative examples. Initially we assume linearly seperable data and later indicate how to handle data which is not linearly seperable.

A natural desideratum is to try to find a decision boundary that maximizes the geometric margin since this would reflect a very confident set of predictions on the training and a good "fit" to the training data. Specifically this results in a clasifier that separates the positive and negative training examples with a gap ("geometric margin"). The method of maximizing this gas is through optimization. It can be shown that the margin width is $2/\|\mathbf{w}\|$.

We have the concept of a margin and now we can formulate the maximum margin classifier. We will first define the hard margin SVM, applicable to a linearly seperable data set, and then modify it to handle non-seperable data. The maximum classifier is the discriminant function that maximizes the geometric margin $1/\|\mathbf{w}\|$ which is equivalent to minimizing $\frac{1}{2}\|\mathbf{w}\|^2$. This leads to the following constrained optimization problem.

$\text{Minimize}_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2$ subject to: $y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1,\ i = 1, \dots, n.$(4)

The constraints in this formulation ensure that the maximum margin classifier classifies each example correctly, which is possible since we assumed that the data is linearly seperable. In practice, data is often not linearly seperable; and even if it is, a greater margin can be achieved by allowing the classifier to misclassify some points. To allow errors we replace the inequality constraints in Eq. (4) with $y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i\ \ i = 1, \dots, n,$ where $\xi_i \geq 0$ are slack variables that allow an example to be in the margin ($0 \leq \xi_i \leq 1$, also called a margin error) or to be misclassified ($\xi_i \geq 1$). Since an example is misclassified if the value of its slack variable is greater than 1, $\sum_i \xi_i$ is a bound on the number of misclassified examples. Our object of maximizing the margin *i.e.*, minimizing $\frac{1}{2}\|\mathbf{w}\|^2$ will be augmented with a term $C \sum_i \xi_i$ to penalize misclassification and margin errors. The optimization problem now becomes

$\text{Minimize}_{w,b} \frac{1}{2}\|w\|^2 + C \sum_{i=1}^n \xi_i$ subject to

$y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i,\ \xi_i \geq 0$
(5)

The constant $C > 0$ sets the relative importance of maximizing the margin and minimizing the amount of slack. This formulation is called the soft margin SVM, and was introduced by Cortes and Vapnik (1995). Using the method of Lagrange multipliers, we can obtain the dual formulation which is expressed in terms of variables α_i (Cotes and Vapnik (1995), ScholKopf and Smola (2002) and Cristianini and Shawe-Taylor (2000)).

$\text{Maximize}_\alpha \sum_{i=1}^n \alpha_i - \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n y_i y_j \alpha_i \alpha_j \mathbf{x}_i^T \mathbf{x}_j$ subject to $\sum_{i=1}^n y_i \alpha_i = 0,\ 0 \leq \alpha_i \leq C$

6. RESULTS AND DISCUSSION

We introduced the Cauchy-Laplace product kernel and using it we trained the support vector machine for the classification of the data using three data sets from the UCI Machine Learning Repository data bases. We have explained that the Cauchy-Laplace product kernel is a Mercer kernel that can overcome the drawbacks of the Laplacian kernel. These factors are well explained in Section 3 and we arrived at the fact that for values of γ small or comparable to 1, the product kernel decreases from 0 faster than the exponential kernel taken alone in a small neighborhood of 0, namely for increasing values of $0 < \|x - x'\| = r < \epsilon$. The results are

shown in the tables given below for various data sets considered for this purpose and illustrate how the product kernel can achieve better classification accuracy. The following table shows the classification results for the diabetic data set

Table 1: The classification results for the diabetic data base

γ value	Gaussian kernel	Laplacian kernel	Product kernel
0.001	72.1345	75.1592	76.5625
0.005	69.2708	74.9469	78.2552
0.01	65.8854	75.1592	77.0833
0.05	65.3646	75.3715	77.7344
0.1	66.2344	75.1592	77.8646
0.5	65.3646	73.4607	77.0833
1.0	65.1042	81.8182	77.2135
5.0	65.1066	80.3922	76.6927

In our recent paper (Chandrasekhar and Akthar 2014), we had compared the Gaussian and Laplacian kernels and noted the better performance of the Laplacian kernel. In the above table, we have one extra column shown for the product kernel. We find that the product kernel has increased the classification performance better than that in the case of the Laplacian for smaller values of γ as we expected under theoretical consideration. However, the Laplacian performs better than the product kernel giving highest accuracy of nearly 82 when γ = 1.When we examine the examples in the diabetic data base, we find that $\|\mathbf{x} - \mathbf{x}'\|$ is rarely very small and therefore even the Laplacian kernel works well. We will find the power of the product kernel in the case of other data sets.

The results are shown in Table 2 below for the experiment conducted on heart data base with 270 examples and 14 attributes.

Table 2: Classification results for the heart data base

γ value	Gaussian kernel	Laplacian kernel	Product kernel
0.001	65.5556	55.5556	81.6667
0.005	60.3704	55.5556	80.8333
0.01	55.5556	55.5556	81.3559
0.05	55.5556	58.5185	78.5124
0.1	60.3212	55.5556	80.6723
0.5	55.5556	58.5185	82.6513
1.0	55.5556	75.9259	82.3529
5.0	55.5444	83.3333	80.3279

We find that the product kernel performed very well, as expected for smaller values of γ. The accuracy is as high as 82.65. However, overall maximum classification 83.33 is achieved by the Laplacian kernel for γ = 5. But it may be case of over fitting. The values beyond 5 are not tabulated as the results are nearly equal. It is to be noted that the classification percentage obtained for smaller values of γ is more reliable than that obtained for larges values of γ for the reasons already stated. There may be over fitting and the model will learn characters that are true about the training examples that do not hold for the overall truth or concept we are trying to learn (i.e., do not generalize). Generalization error is the excess error rate we observe when scoring new examples versus the error rate we saw in learning the training data. So we cannot expect generalization using accuracy obtained for larger γ. In this way, the product kernel is significant in predicting the classification better. The accuracy determined is uniform and consistent with respect to values of γ.

When γ is small or comparable to 1, a given data point **x** has a non-zero kernel value relative to any example $\mathbf{x}'$ in the set of support vectors. Therefore the whole set of support vectors will affect the value of the discriminant function at **x**, resulting in a smooth decision boundary. As γ increases, as described in the above paragraph, the locality of the support vector expansion decreases, leading to greater curvature of the decision boundary. As a result, the hyper plane can be bumpy and over fitting can occur.

The classification results for the C M C data base with 3772 vectors and 22 attributes are shown in Table 3.

Table 3: Classification results for the C M C data base

γ value	Gaussian kernel	Laplacian kernel	Product kernel
0.001	51.1881	57.1881	67.1419
0.005	54.8540	57.5017	67.6740
0.01	54.9219	57.4338	67.0120
0.05	55.3293	57.7733	67.2777
0.1	55.6008	56.4834	66.5309
0.5	52.8853	63.4759	66.0557
1.0	48.5404	66.9382	66.8703
5.0	43.3809	66.7217	66.6667

While the Gaussian kernel gives an accuracy of 55.6 for $\gamma = 0.1$, the Laplacian kernel gives 66.93 for $\gamma = 1$. The product kernel gives an accuracy of 67.67 for smaller values of γ. Though the accuracy given by the product kernel is the maximum among the three, the important information to note is that it is achieved for smaller values of γ. Since the support vector machine gives smoother hyper plane for smaller values of γ, the result obtained using the product kernel is more reliable than that obtained using the Laplacian kernel which gives the maximum accuracy only for larger values of γ. It is likely that over fitting may have arisen. So, it is safer to consider the classification result given by the product kernel.

7. CONCLUSION

It is shown that the Cauchy-Laplace product kernel brings out better classification performance over the Laplacian kernel in classifying data using a support vector machine. The important information to note is that better classification is achieved for smaller values of kernel parameterγ. Since the support vector machine gives smoother hyper plane for smaller values of γ, the result obtained using the product kernel is more reliable than that obtained using the Laplacian

kernel which gives the maximum accuracy only for larger values of γ and it is likely that over fitting may have arisen. So, it is safer to consider the classification result given by the product kernel.

8. REFERENCES

[1] Aizerman, M. A., Braverman, E. M. and Rozonoer, L. I. Theoretical foundations of the potential function method in pattern recognition learning, Automation and Remote Control, Vol. 25, p.821–837, 1964.

[2] Basak, Jayanta. A least square kernel machine with box constraints, International Conference on Pattern Recognition 2008, 1, p.1-4, 2008.

[3] Chandrasekhar, P. and Akthar, P. Md. Advantages of using Laplacian kernel over Gaussian RBF in a Support Vector Machine, International Journal of Merging Technology Advanced Research in Computing, Issue IV, Vol.1, ISSN-2320-1363, Dec.2013.

[4] Cortes, C. and Vapnik, V. Support vector networks, Machine Learning, Vol. 20, p.273-297,1995.

[5] Cristianini, N. and Shawe-Taylor, J. An Introduction to Support Vector Machines, Cambridge University Press, Cambridge, uk,2000.

[6] Girosi, F. An equivalence between sparse approximation and support vector machines. Neural Computation, Vol. 20, p.1455–1480,1998.

[7] Girosi, F., Jones, M. and Poggio, T. Regularization theory and neural network architectures. Neural Computation, Vol. 7, p.219–269,1995.

[8] Mercer, J. Functions of positive and negative type and their connection with the theory of integral equations, Transactions of the London Philosophical Society (A), Vol. 209, pp. 415-446,1909.

[9] Sangeetha, R. and Kalpana, B. Performance Evaluation of Kernels in multiclass Support Vector Machines, International Journal of Soft Computing and Engineering, (IJSCE), ISSN: 2231-2307, Vol. 1, Issue 5, p.138-145, November,2011.

[10] Sch”olkopf, B. and A. Smola, A. Learning with Kernels, MIT Press, Cambridge, MA,2002.

[11] Smola, A. J., Scho¨lkopf, B. and Mu¨ller, K. R. The connection between regularization operators and support vector kernels, Neural Networks, Vol. 11, p.637–649, 1998.