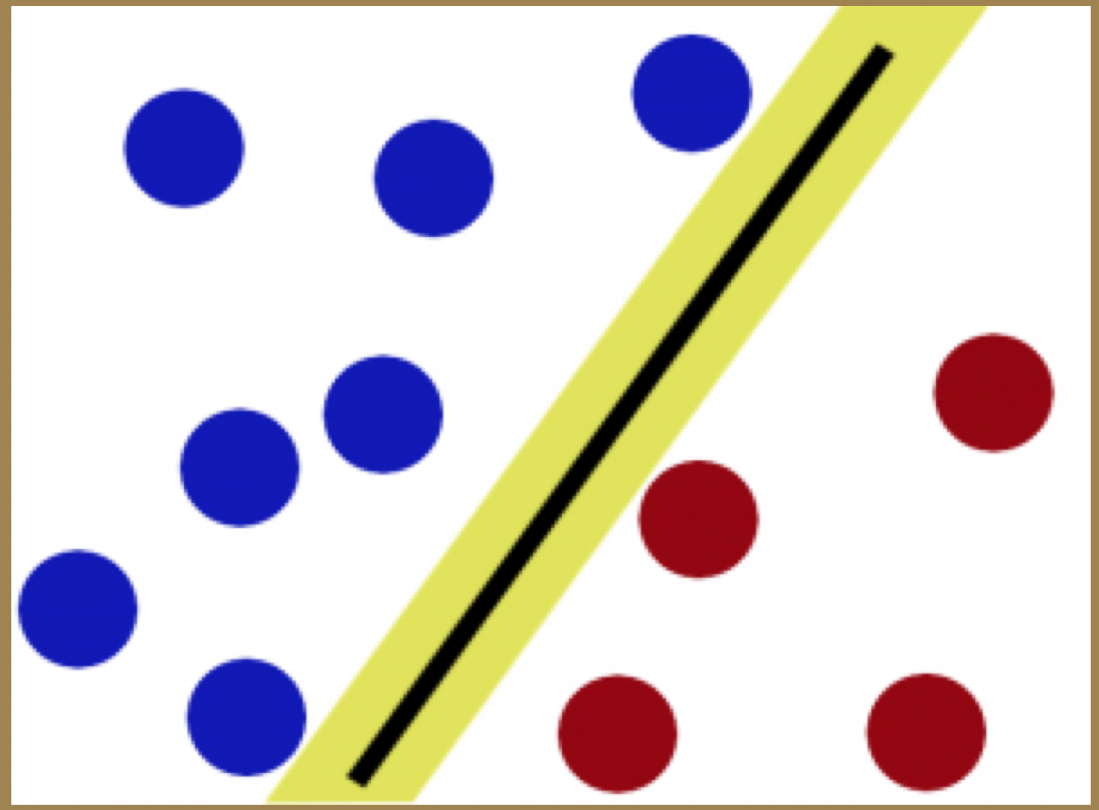
Support Vector Machines



Presented by Ramya Balasubramaniam

Introduction: Simple Idea

Simple idea behind proposing Support Vector Machines: To map the input vectors belonging to a lower dimensional space say n, to a high dimensional feature space say N without increasing the computational and the space complexity of the problem much.

Main Highlights of this paper with regards to the effectiveness of SVMs:

- SVMs were capable of classifying both linearly separable and non-separable data points with minimum errors (as compared to any contemporary algorithm).

- SVMs had a very high generalization ability, with non-linear transformations that were independent of input dimension size and training data size.

Introduction: Evolution of Algorithms

In 1930s R.A Fisher came up with the ideas of Linear and Quadratic Discriminant analyses for data points that could be separated using linear and quadratic decision boundaries.

Basic Assumptions : The model consisted of two populations that were normally distributed $N(m_1, \Sigma_1)\ and\ N(m_2, \Sigma_2)$ of n dimensional vectors x with mean vectors $m_1\ and\ m_2$ co-variate matrices $\Sigma_1 and\ \Sigma_2$,.

The optimal Bayesian solution is a quadratic decision boundary if $\Sigma_1 \neq \Sigma_2$ and is given by:

$$F_{sq}(\mathbf{x}) = \text{sign}\left[\frac{1}{2}(\mathbf{x}-\mathbf{m}_1)^T\boldsymbol{\Sigma}_1^{-1}(\mathbf{x}-\mathbf{m}_1) - \frac{1}{2}(\mathbf{x}-\mathbf{m}_2)^T\boldsymbol{\Sigma}_2^{-1}(\mathbf{x}-\mathbf{m}_2) + \ln\frac{|\boldsymbol{\Sigma}_2|}{|\boldsymbol{\Sigma}_1|}\right]. \quad (1)$$

If $\Sigma_1 = \Sigma_2$, we get a linear decision boundary that is given by:

$$F_{lin}(\mathbf{x}) = \text{sign}\left[(\mathbf{m}_1-\mathbf{m}_2)^T\boldsymbol{\Sigma}^{-1}\mathbf{x} - \frac{1}{2}\left(\mathbf{m}_1^T\boldsymbol{\Sigma}^{-1}\mathbf{m}_1 - \mathbf{m}_2^T\boldsymbol{\Sigma}^{-1}\mathbf{m}_2\right)\right].$$

Introduction: Evolution of Algorithms

To compute a quadratic decision boundary, one needed to determine $\frac{n(n+3)}{2}$ parameters

To compute a linear decision boundary, one needed to determine n parameters.

Fisher proposed that even when $\Sigma_1 \neq \Sigma_2$, we could use linear decision boundary by computing a common Σ of the form:

$$\boldsymbol{\Sigma} = \tau\boldsymbol{\Sigma}_1 + (1-\tau)\boldsymbol{\Sigma}_2,$$

Fisher went on to propose a linear decision function for cases where the two distributions were not normal.

All the initial algorithms proposed for pattern recognition constructed linear decision boundaries.

Introduction: Evolution of Algorithms

In 1960s Rosenblatt came up with neural networks: Perceptron which consisted of connected neurons, where each neuron implemented a separating hyperplane, the perceptron was thus a piecewise linear separating surface as shown below:

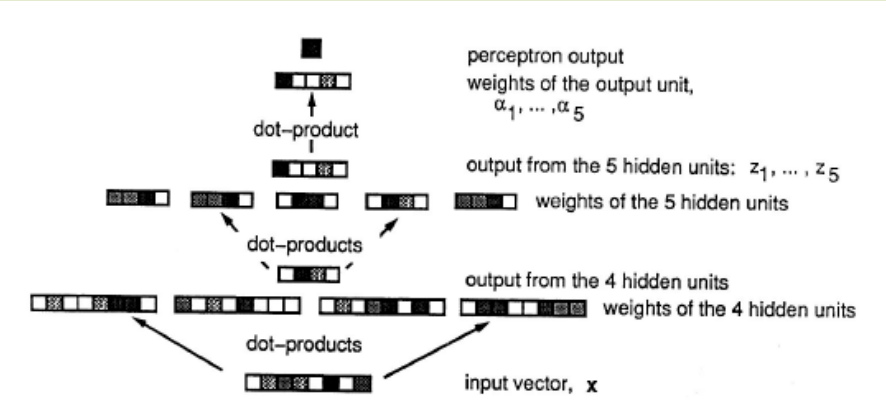


Figure 1. A simple feed-forward perceptron with 8 input units, 2 layers of hidden units, and 1 output unit. The gray-shading of the vector entries reflects their numeric value.

Unlike the present day neural networks, only the weights of the output were adaptive. The input vectors are non-linearly transformed into feature space Z, of last layer of units. A linear decision function is constructed:

$$I(\mathbf{x}) = \text{sign}\left(\sum_i \alpha_i z_i(\mathbf{x})\right)$$

In late 1980s, it was Rumelhart, Hinton and Williams who came up with the back-propagation method, so all the weights of the neural network became adaptive.

Introduction: Why SVM?

To obtain a decision boundary surface corresponding to a polynomial of degree two, one can create a feature space Z, which has $=\frac{n(n+3)}{2}$ parameters , as shown below

$$
\begin{aligned}
z_1 = x_1, \ldots, z_n = x_n, &\qquad n \text{ coordinates,} \\
z_{n+1} = x_1^2, \ldots, z_{2n} = x_n^2, &\qquad n \text{ coordinates,} \\
z_{2n+1} = x_1 x_2, \ldots, z_N = x_n x_{n-1}, &\qquad \frac{n(n-1)}{2} \text{ coordinates,}
\end{aligned}
$$

where $\mathbf{x} = (x_1, \ldots, x_n)$. The hyperplane is then constructed in this space.

Two problem with this approach:

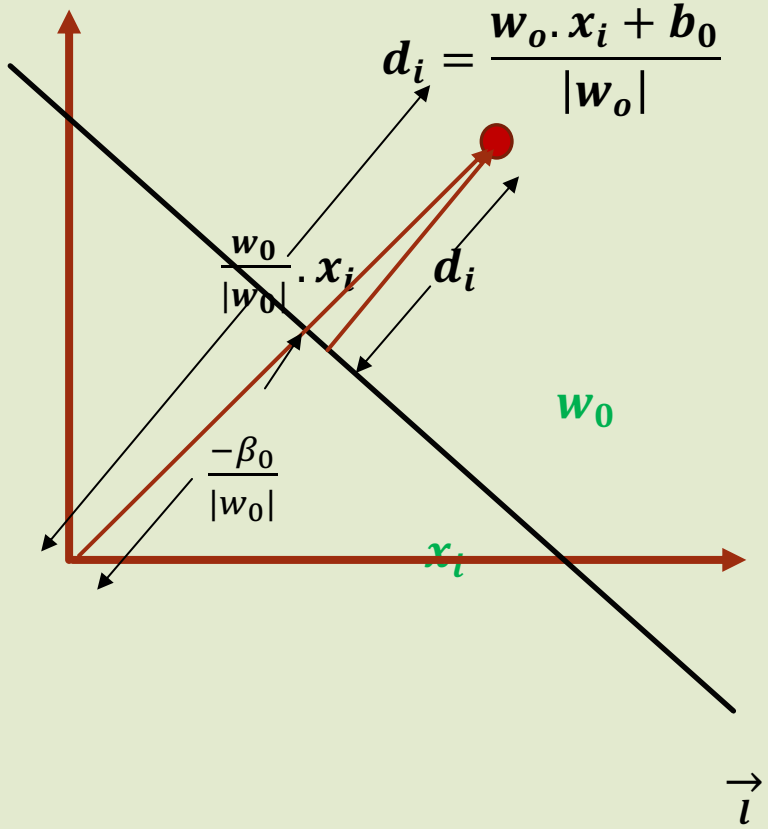
- Conceptual Problem: Given the dimensionality of feature space, how could we find a separating plane that could generalize well?
- Technical Problem: How computationally would we treat high-dimensional space?

The conceptual problem was solved by defining class of hyperplanes that had maximum margin between vectors of the two classes. In order to construct this hyperplane one had to remember only the support vectors that defined the decision hyperplane. The average probability of error was given by:

$$
E[\Pr(\text{error})] \leq \frac{E[\text{number of support vectors}]}{\text{number of training vectors}}.
$$

Hard Margin Support Vector Machines

So if we have a point represented by x_i and vector w_0 represents the direction perpendicular to a given line l , - $\frac{\beta_0}{|w_0|}$ is distance of the line from origin, then, the distance d_i of the point from line l is:



So, for any point on the line l, this distance $d_i = 0$

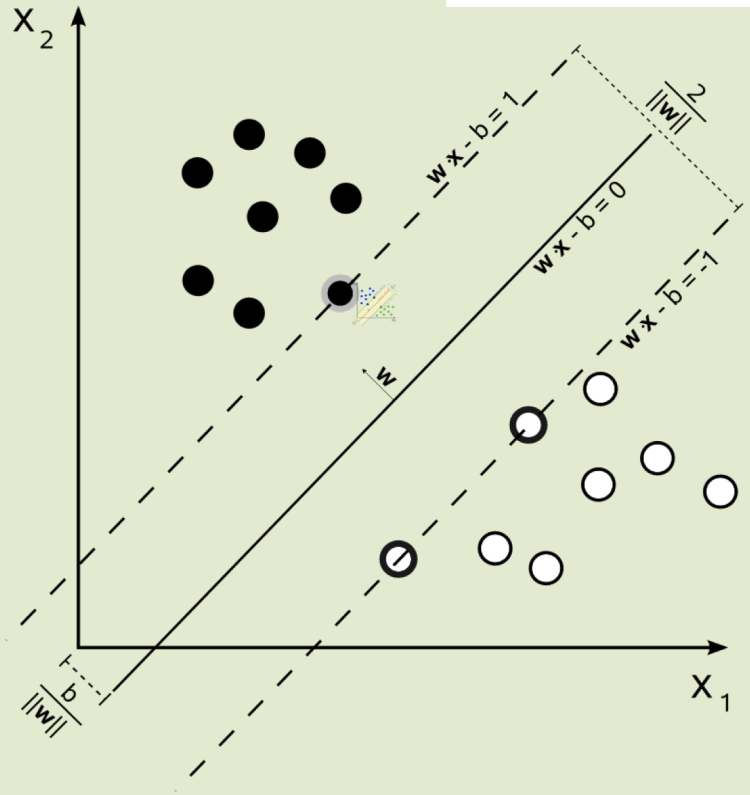
Hard Margin Support Vector Machines

When we have points in a dataset (y_1,x_1), (y_2,x_2) .... (y_l,x_l) that are linearly separable, as shown in the figure, we can represent points belonging to two separate classes (on either side of the line) as

$$\begin{aligned} \mathbf{w}\cdot\mathbf{x}_i + b &\geq 1 \quad \text{if} \quad y_i = 1, \\ \mathbf{w}\cdot\mathbf{x}_i + b &\leq -1 \quad \text{if} \quad y_i = -1, \end{aligned}$$

which can be represented by the equation

$$y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1, \qquad i = 1, \ldots, \ell. \tag{1}$$



Hard Margin Support Vector Machines

So, margin between two points belonging to two different classes is given by

$$\rho(\mathbf{w}_0, b_0) = \frac{2}{|\mathbf{w}_0|} = \frac{2}{\sqrt{\mathbf{w}_0 \cdot \mathbf{w}_0}}.$$

If $\rho(\mathbf{w}, b)$ represents the margin and the optimal hyperplane separating the two groups of points is $\mathbf{w}_0 \cdot \mathbf{x} + b_0 = 0$

In order to maximize $\rho(\mathbf{w}, b)$ we would turn this problem to one where we minimize $\mathbf{w} \cdot \mathbf{w}$

For this we make use of the following Lagrangian:

$$\boldsymbol{L(w, \beta_0, \alpha_i) = \frac{1}{2}w^2 - \Sigma\alpha_i\left(y_i\left(w^T . x_i + \beta_0\right) - 1\right)} \quad \textbf{(2)}$$

where each $\boldsymbol{\alpha_i}$ is just a Lagrange multiplier in the Lagrangian and $\boldsymbol{\alpha_i \geq 0}$

Hard Margin Support Vector Machines

$$L(w, \beta_0, \alpha_i) = \frac{1}{2}|w|^2 - \Sigma\alpha_i\left(y_i\left(w^T.x_i + \beta_0\right) - 1\right) \textbf{(2)}$$

By differentiate L w.r.t w (and making it equal to 0) we get: $\Sigma_i \alpha_i y_i x_i = w$

By differentiating L w.r.t $\boldsymbol{\beta_0}$ we get: $\Sigma_i \alpha_i y_i = 0$

When we substitute the value of w obtained above in equation (2) we get:

$$L(w, \beta_0, \alpha_i) = \frac{1}{2}\Sigma_i\Sigma_j\alpha_i\alpha_j x_i x_j y_i y_j - \Sigma_i\Sigma_j\alpha_i\alpha_j x_i x_j y_i y_j - \Sigma_i\alpha_i y_i\beta_0 + \Sigma\alpha_i \quad (\Sigma\alpha_i y_i\beta_0 = 0)$$

$$= \Sigma\alpha_i - \frac{1}{2}\Sigma_i\Sigma_j\alpha_i\alpha_j(y_i y_j\, x_i.x_j)$$

From this we get the Lagrangian in terms of the dual variable/parameters. So, now we solving the following quadratic problem (of maximizing w.r.t the Lagrange multiplier)

$$\Lambda^T = (\alpha_1, \ldots, \alpha_\ell).$$

$$W(\Lambda) = \Lambda^T \mathbf{1} - \frac{1}{2}\Lambda^T \mathbf{D}\Lambda$$

where $\mathbf{1}^T = (1, \ldots, 1)$ is an ℓ-dimensional unit vector, $\mathbf{Y}^T = (y_1, \ldots, y_\ell)$ is the ℓ-dimensional vector of labels, and $\mathbf{D}$ is a symmetric $\ell \times \ell$-matrix with elements

And subject to the constraints:

$$\Lambda \geq 0,$$
$$\Lambda^T \mathbf{Y} = 0,$$

Soft Margin Support Vector Machines

When the points in data set are not separable then the following changes need to be made to the constraint (1)

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i, \qquad i = 1, \ldots, \ell,$$
$$\xi_i \geq 0, \qquad i = 1, \ldots, \ell.$$

We define a function associated with this error term that must be minimize, $\Phi(\xi) = \sum_{i=1}^{\ell} \xi_i^{\sigma}$

Apart from vector w we also minimize the above function, so our function to we minimize

$$\frac{1}{2}\mathbf{w}^2 + CF\left(\sum_{i=1}^{\ell} \xi_i^{\sigma}\right)$$

Where f(u) is a monotonic convex function and C is a constant.

End objective -> we minimize the number of errors on the training set and separate the rest of the points with maximum margin. (C sufficiently large and σ *is small*.

To avoid making this problem NP complete, the authors propose $\sigma = 1$

This reduces $\varphi(\xi)$ to sum of deviations of training errors.

Soft Margin Support Vector Machines

The modifications to the Lagrangian:

$$L(w, \beta_0, \xi_i, \alpha_i, \lambda_i) = \frac{1}{2}w^2 + \delta \sum_i \xi_i - \Sigma\alpha_i\left(y_i\left(w^T . x_i + \beta_0\right) - 1 - \xi_i\right) - \Sigma\lambda_i\xi_i \quad (3)$$

After differentiating this w.r.t w we get

$$w = \Sigma\alpha_i x_i y_i$$

differentiating this w.r.t β_0 we get

$$\Sigma\alpha_i y_i = 0$$

differentiating this w.r.t ξ_i we get

$$(\alpha_i + \lambda_i) = \delta$$

Substituting these values in equation 3 we get the following

$$\sum \alpha_i - \frac{1}{2}\sum\sum \alpha_j \alpha_i x_i x_j y_j y_i$$

If $(\alpha_i + \lambda_i) = \delta$, and $\lambda_i \geq 0$, $then\ \delta \geq \alpha_i$

This has been represented in the paper by:

$$\begin{aligned} \Lambda^T \mathbf{Y} &= 0, \\ \delta &\geq 0, \\ 0 \leq \Lambda &\leq \delta 1, \end{aligned}$$

Method of Convolution of Dot Products in Feature Space

Usual approach to transform data from input space to feature space is:

$$\phi: \Re^n \to \Re^N.$$

Then compute an N dimensional linear separator w and a bias b, based on transformed vectors

$$\phi(\mathbf{x}_i) = \phi_1(\mathbf{x}_i), \phi_2(\mathbf{x}_i), \ldots, \phi_N(\mathbf{x}_i), \qquad i = 1, \ldots, \ell.$$

So, now our classification of vector x depends on the function: $f(\mathbf{x}) = \mathbf{w} \cdot \phi(\mathbf{x}) + b.$

And we know that $\mathbf{w} = \sum_{i=1}^{\ell} y_i \alpha_i \phi(\mathbf{x}_i).$ and with linearity of dot-products, the classification function given by:

$$f(\mathbf{x}) = \phi(\mathbf{x}) \cdot \mathbf{w} + b = \sum_{i=1}^{\ell} y_i \alpha_i \phi(\mathbf{x}) \cdot \phi(\mathbf{x}_i) + b.$$

This means that the classification function only depends on dot products between transformed input vectors

Method of Convolution of Dot Products in Feature Space

So, we need not compute $\varphi(x)$ and then, classification function.

We use a kernel function that simultaneously transforms the input and calculates the dot product in a Hilbert Space.

$$\phi(\mathbf{u}) \cdot \phi(\mathbf{v}) \equiv K(\mathbf{u}, \mathbf{v}).$$

This K(u,v) can be expanded as

$$K(\mathbf{u}, \mathbf{v}) = \sum_{i=1}^{\infty} \lambda_i \phi_i(\mathbf{u}) \cdot \phi_i(\mathbf{v}),$$

λ_i are eigenvalues (all must be positive) and φ_i are eigen functions, for the eigenvalues to be positive, K(u,v) must satisfy Mercer's equation for all g where $\int g^2(\mathbf{u})d\mathbf{u} < \infty.$ such that :

$$\int K(\mathbf{u}, \mathbf{v})\phi_i(\mathbf{u})d\mathbf{u} = \lambda_i \phi_i(\mathbf{v}).$$

Using different kernel functions different support networks can be built. So, basically do:

$$f(\mathbf{x}) = \sum_{i=1}^{\ell} y_i \alpha_i K(\mathbf{x}, \mathbf{x}_i),$$

General Features of Support Vector Machines

I. Constructing support vector machine is efficient

A dual quadratic optimization problem of solving

$$W(\Lambda) = \Lambda^T \mathbf{1} - \frac{1}{2}\left[\Lambda^T \mathbf{D}\Lambda + \frac{\delta^2}{C}\right],$$

under the simple constraints:

$$\mathbf{0} \leq \Lambda \leq \delta \mathbf{1},$$
$$\Lambda^T \mathbf{Y} = 0,$$

where matrix

$$D_{ij} = y_i y_j K(\mathbf{x}_i, \mathbf{x}_j), \qquad i, j = 1, \ldots, l.$$

Can be optimally computed by making use of the intermediate approach to solving proposed by the author instead of taking all training data at once.

General Features of Support Vector Machines

II. Support Vector Machine is Universal Machine

With the introduction of the concept of convolution of dot product, any kernel function could be used to design decision boundaries to separate both linearly separable as well as non-separable training sets.

Polynomial kernel function:

$$K(\mathbf{u}, \mathbf{v}) = (\mathbf{u} \cdot \mathbf{v} + 1)^d.$$

Radial Basis Function:

$$K(\mathbf{u}, \mathbf{v}) = \exp\left\{-\frac{|\mathbf{u} - \mathbf{v}|^2}{\sigma^2}\right\}$$

General Features of Support Vector Machines

III. Support Vector Machines control Generalization Ability:

Testing error is associated with a generalization error bound given by:

$$R_{test} \leq R_{training} + \sqrt{\frac{VC(H)\log\frac{2n}{VC(H)} - \log\frac{\delta}{4}}{n}}$$

VC(H) is Vapnik-Chervonenkis dimension, δ is error probability, n is training set size

For support vector machine, if $|x| < R$, VC dimension is given by

$$R^2 \; W^T \, W$$

Support Vector Machine, we minimize the function W.W and control R such that the training error is also minimized.

Pr(test error) $\leq$ Frequency(training error) + Confidence Interval

Experimental Analysis

Two types of experiments were conducted by authors to demonstrate support vector network

Experiment 1: Constructed artificial sets of patterns in the plane and experiment with second order polynomial decision boundary.

Experiment 2: Conducted experiments with the real-life problem of digit recognition

Experimental Analysis: Experiment 1

Task performed:

- Authors constructed decision rules for different set of patterns in the plane using the following second order polynomial Kernel function, given by:

$$K(\mathbf{u}, \mathbf{v}) = (\mathbf{u} \cdot \mathbf{v} + 1)^d$$

where d = 2

- Results of the experiment were visualized to illustrate the power of this algorithms

Experimental Analysis: Experiment 1

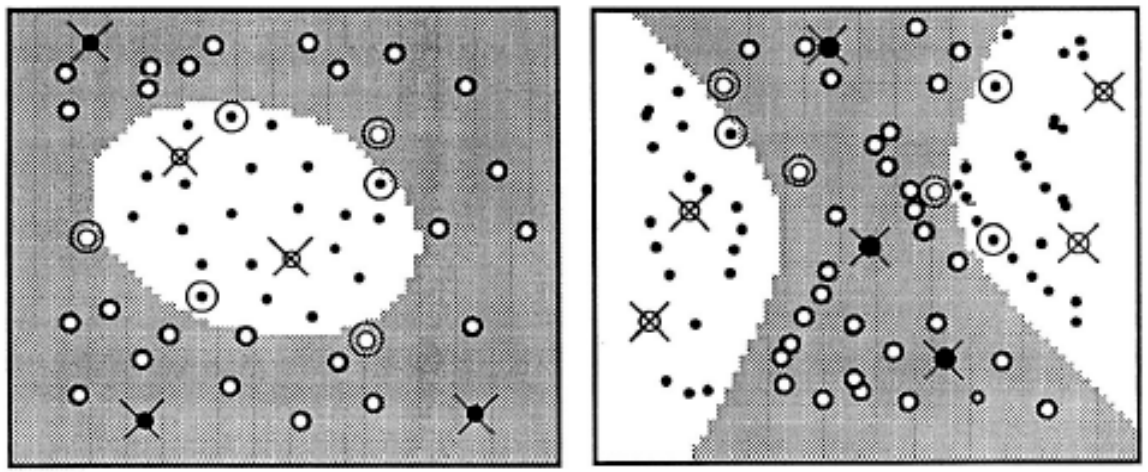


Figure 5. Examples of the dot-product (39) with $d = 2$. Support patterns are indicated with double circles, errors with a cross.

Observations:

- The 2 classes were represented by white and black bullets
- The double circle indicated the support patterns
- The crosses indicate the errors

Authors claim that the given solution is the most optimal solution in terms of any 2^{nd} degree polynomial: No other polynomial could commit fewer errors in classification

Experimental Analysis: Experiment 2

Task performed:

- Authors took two different types of databases for bit-mapped digit recognition, a small and large database.
  - **USPS database:** The small database consisted of 7,300 training and 2,000 test patterns, 16X16 resolution, authors reported experimental research with polynomials of various degrees.
  - **50-50 mixture of NIST database:** The large database consisted of 60,000 training and 10,000 test patterns, 28X28 resolution, authors constructed only a 4^{th} degree polynomial classifier. Performance compared to other types of learning machines such as Decision trees(CART, C4.5),2-layer and 5-layer Neural networks and humans.
- In all experiments ten separators were constructed, one for each class (10 digits), same dot products and pre-processing of data used.

Experimental Analysis: Experiment 2

For the USPS Database:

List of performance of various classifiers collected from publications and own experiment

Table 1. Performance of various classifiers collected from publications and own experiments. For references see text.

Classifier	Raw error, %
Human performance	2.5
Decision tree, CART	17
Decision tree, C4.5	16
Best 2 layer neural network	6.6
Special architecture 5 layer network	5.1

Pre-processing used: Centering, de-slanting and smoothing, input dimension of 256, order of polynomials used 1 to 7, training data not linearly separable.

Experimental Analysis: Experiment 2

For the USPS Database: Results

- The number of support vectors increases very slowly with the degree of polynomial chosen unlike the dimensionality that increases phenomenally.
- The performance almost does not change with increasing dimensionality of space- No over-fitting problem
- From the linear case to 2nd degree polynomial, the number of misclassifications falls from 34 to just 4 per classifier (on an average)
- In all the cases, the bound on generalization ability holds, upper bound on error probability does not exceed 3%
- Training time for construction of polynomial classifiers does not depend on degree of polynomial

Table 2. Results obtained for dot products of polynomials of various degree. The number of "support vectors" is a mean value per classifier.

Degree of polynomial	Raw error, %	Support vectors	Dimensionality of feature space
1	12.0	200	256
2	4.7	127	~33000
3	4.4	148	$\sim 1 \times 10^6$
4	4.3	165	$\sim 1 \times 10^9$
5	4.3	175	$\sim 1 \times 10^{12}$
6	4.2	185	$\sim 1 \times 10^{14}$
7	4.3	190	$\sim 1 \times 10^{16}$

Experimental Analysis: Experiment 2

For the NIST database:

Used for benchmark studies conducted over just two weeks – only one type of classifier was constructed- 4th degree polynomial with no pre-processing (choice based on US Postal database)

Table 3. Results obtained for a 4th degree polynomial classifier on the NIST database. The size of the training set is 60,000, and the size of the test set is 10,000 patterns.

	Cl. 0	Cl. 1	Cl. 2	Cl. 3	Cl. 4	Cl. 5	Cl. 6	Cl. 7	Cl. 8	Cl. 9
Supp. patt.	1379	989	1958	1900	1224	2024	1527	2064	2332	2765
Error train	7	16	8	11	2	4	8	16	4	1
Error test	19	14	35	35	36	49	32	43	48	63

Observations:

- Even a 4th degree polynomial commit errors on training set
- Upper bound on errors is still valid, combined test error rate 1.1%
- SVM results were compared with other participating classifiers

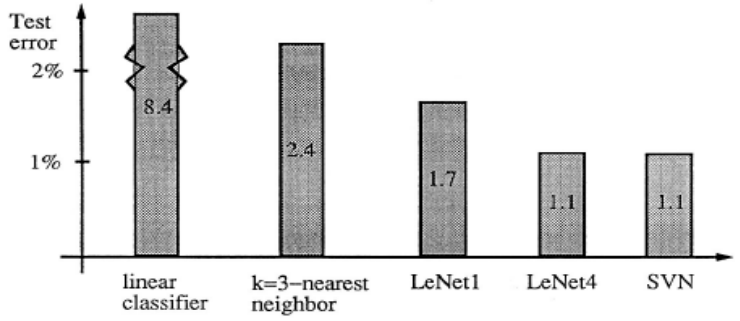


Figure 9. Results from the benchmark study.

Conclusion

- The paper introduced SVN as a learning machine for two-group classification problems.

- The support vector network combines 3 ideas:
  - Solution technique from optimal hyperplanes (that allows expansion pf solution w.r.t support vectors)
  - Idea of convolution of Dot-products (extending the solution to non-linear cases)
  - Notion of soft-margin (allows errors on training set)