CSE-E4810 Machine Learning and Neural Networks (5 cr)

Lecture 8: Support Vector Machines

Prof. Juha Karhunen

https://mycourses.aalto.fi/course/view.php?id=13086

Introduction

- This lecture is based on Chapter 6 "Support Vector Machines" in the book S. Haykin, "Neural Networks - A Comprehensive Foundation", 2nd ed., Prentice-Hall, 1998.
- Support vector machines (SVMs) constitute an important machine learning and neural networks technique with many applications.
- They are discussed also in Chapter 16 in Du's and Swamy's book.
- The discussion there is less thorough than in Haykin's book but contains many extensions and modifications of SVM's
- The page `http://www.support-vector-machines.org/` contains links to a wealth of material (books, software, etc.) on SVMs.

Properties of Support Vector Machines (SVMs)

- A straightforward engineering solution for classification tasks.
- Support vector machines have nice computational properties.
- Key ideas in brief (note: several intertwined ideas $\Rightarrow$ hard to digest):
  - Construct a separating hyperplane in a high-dimensional feature space.
  - Maximize separability.
  - Express the hyperplane in the original space using a small set of training vectors, the "support vectors".
  - SVMs are nonlinear in the input space.
- Yet another universal feedforward network type.
- Suitable tasks: pattern classification, nonlinear regression.
- Have a connection to structural risk minimization (VC dimension).

Lecture plan

- Understanding support vector machines requires lots of new stuff, so let us proceed bit by bit.

1. **Pattern classification** (Sections 6.2 - 6.6)
   (a) "Trivial" linear SVMs (Sections 6.2 - 6.3)
   - Separable classes
   - Connection to VC (Vapnik-Chervonenkis) dimension
   - Nonseparable classes
   - Computation of the solution

   (b) Nontrivial nonlinear SVM:s (Section 6.4)
   - "Kernel trick"

   (c) Examples (Sections 6.5 - 6.6)
2. **Nonlinear regression** (Sections 6.7 - 6.8)

Optimal hyperplane for linearly separable classes

- **Goal:** Construct a separating hyperplane that **maximizes the margin of separation.**
- For simplicity, we consider first two pattern classes which are **linearly separable.**
- We have a set of N training samples $\{\mathbf{x}_i, d_i\}_{i=1}^N$.
- The desired response is $d_i = +1$ for vectors $\mathbf{x}_i$ belonging to the first class and $d_i = -1$ for the training vectors in the second class.
- One can construct a linear decision surface (hyperplane) which separates the classes:

$$\begin{aligned} \mathbf{w}^T\mathbf{x}_i + b &\geq 0 \text{ for } d_i = 1 \\ \mathbf{w}^T\mathbf{x}_i + b &< 0 \text{ for } d_i = -1 \end{aligned} \tag{1}$$

- Consider a given weight vector $\mathbf{w}$ and bias b.
- The separation ρ between the hyperplane

$$\mathbf{w}^T\mathbf{x} + b = 0 \tag{2}$$

and the closest data point is called the **margin of separation.**

- The goal of a support vector machine is to find the optimal hyperplane for which the margin of separation ρ is maximized.
- The geometric construction of an optimal hyperplane is illustrated in Figure 1 for a two-dimensional case.
- Denote by $\mathbf{w}_o$ and b_o the optimal values for the weight vector $\mathbf{w}$ and the bias b.

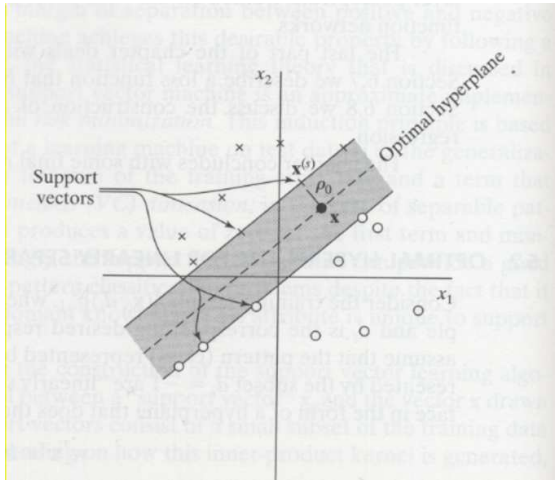

Figure 1: Optimal hyperplane for linearly separable patterns.

- Note that the scale of $\mathbf{w}$ and b is arbitrary:

$$c\mathbf{w}^T\mathbf{x} + cb = 0 \tag{3}$$

yields the same decision surface.

- Hence we can fix the scale by setting the distance from the surface to the closest sample on either side to unity:

$$\begin{aligned} \mathbf{w}_o^T\mathbf{x}_i + b_o &\geq 1 \text{ for } d_i = 1 \\ \mathbf{w}_o^T\mathbf{x}_i + b_o &\leq -1 \text{ for } d_i = -1 \end{aligned} \tag{4}$$

- **Support vectors** are the vectors that are closest to the decision surface:

$$\mathbf{w}_o^T\mathbf{x}_i + b_o = \pm 1 \tag{5}$$

- Support vectors are those data points that lie closest to the decision surface, and they are therefore the most difficult to classify.

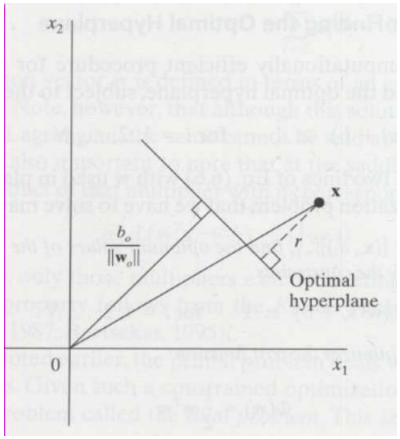

Figure 2: Geometric interpretation of algebraic distances of points to the optimal hyperplane for a two-dimensional case.

Maximizing the margin of separation

- Decompose a support vector $\mathbf{x}$ as

$$\mathbf{x} = \mathbf{x}_p + r\frac{\mathbf{w}_o}{\|\mathbf{w}_o\|} \tag{6}$$

- Here $\mathbf{x}_p$ is the normal projection of $\mathbf{x}$ onto the optimal hyperplane.
- And r is the distance of the support vector $\mathbf{x}$ from the optimal separating hyperplane.
- See Figure 2 for a geometric illustration of the situation.
- After some fairly simple algebraic manipulations, one can derive the result

$$r = \frac{1}{\|\mathbf{w}_o\|} \tag{7}$$

- See Haykin's book, pp. 320-322 for mathematical details.

- Hence, the maximal margin of separation ρ is

$$\rho = \frac{2}{\|\mathbf{w}_o\|} \tag{8}$$

- Thus maximizing the margin of separation between the classes is equivalent to minimizing the Euclidean norm of the weight vector $\mathbf{w}$.
- The optimal hyperplane is **unique,** defined by the optimal weight vector $\mathbf{w}_o$.

Connection to structural risk minimization

- Support vector machines can be justified by results from learning theory.
- More precisely, SVM is an approximate implementation of the method of **structural risk minimization.**

- The principle of structural risk minimization says that the generalization error of a learning machine for test data is bounded by the sum of:
  - Training error rate;
  - And a term that depends on the **Vapnik-Chernovenkis (VC) dimension.**
- VC dimension measures the capacity of a classifier.
- It can be shown that the VC-dimension h is bounded by a term that decreases as the margin of separation ρ increases.
- Here training error $= 0$ because the patterns are linearly separable.
- Hence we should **use the hyperplane for which ρ is maximal.**
- Maximizing the margin of separation ρ minimizes the structural risk.

- Structural risk minimization and VC dimension are discussed in Section 2.14 in Haykin's book mentioned in the beginning and in Section 2.8 in Du's and Swamy's book.
- We skip their discussion in our course for two major reasons:
  - This learning theory is a highly theoretical topic which is difficult to understand.
  - The bounds provided by Vapnik-Chernovenkis dimension for the number of samples required for achieving a certain performance level are of little practical value.
  - This is because they are overly pessimistic, based on worst case considerations.

How to compute the optimal hyperplane

- **Goal:** Find the weight vector $\mathbf{w}$ having the smallest norm and fulfilling the following constraint for each sample pair $(\mathbf{x}_i, d_i)$:

$$d_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \qquad i = 1, 2, \ldots, N \tag{9}$$

  (This is a shorthand notation; note that $d_i = \pm 1$.)
- Now the cost function $\phi(\mathbf{w}) = \mathbf{w}^T\mathbf{w}/2$ is convex.
- And the constraints (9) are linear in $\mathbf{w}$.
- From optimization theory, the problem can then be solved using the **method of Lagrange multipliers.**
- First construct the Lagrangian function

$$J(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{N} \alpha_i[d_i(\mathbf{w}^T\mathbf{x}_i + b) - 1] \tag{10}$$

  - And then minimize it with respect to weight vector $\mathbf{w}$ and bias b.
  - And maximize it with respect to **Lagrance multipliers** α_i, $i = 1, 2, \ldots, N$.
- It is an old result in optimization theory that the problem has a **dual problem** that has an equivalent solution.
- The dual problem is obtained by (details in the book):
  - Solving for the zeros of the derivative of J w.r.t $\mathbf{w}$ and b
  - Inserting the resulting values of $\mathbf{w}$ and b to J
- The resulting objective function is

$$Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i\alpha_j d_i d_j \mathbf{x}_i^T\mathbf{x}_j \tag{11}$$

- The function Q should be maximized w.r.t α, subject to the constraints

1. $\sum_{i=1}^{N} \alpha_i d_i = 0$
2. $\alpha_i \geq 0$

- Finally, the optimum decision function (separating hyperplane) is

$$\mathbf{w}_o^T\mathbf{x} + b_o \tag{12}$$

- There the optimum weight vector

$$\mathbf{w}_o = \sum_{i=1}^{N} \alpha_{o,i} d_i \mathbf{x}_i \tag{13}$$

  where the coefficients $\alpha_{o,i}$ are nonzero only for the support vectors!
- The optimum bias b_o is given by Eq. (6.18) in Haykin's book.
- There is a lot of free software available on SVMs $\Rightarrow$ you need not know all the technical details of optimization.

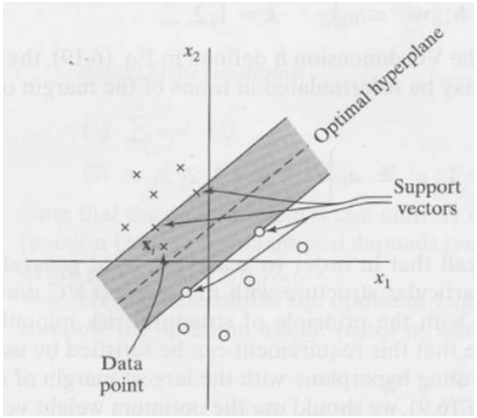

Figure 3: Data point $\mathbf{x}_i$ (belonging to class $\mathcal{C}_1$) falls inside the region of the separation but on the right side of the decision surface.

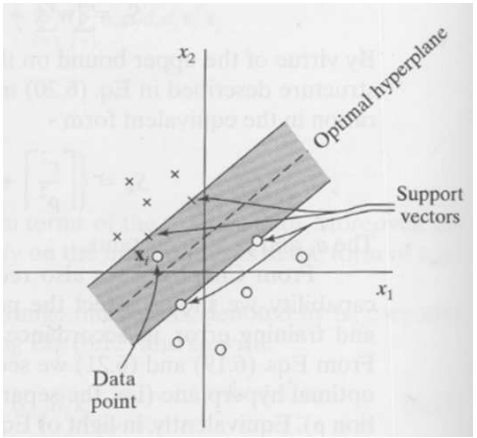

Figure 4: Data point $\mathbf{x}_i$ (belonging to class $\mathcal{C}_2$) falls on the wrong side of the decision surface.

Optimal hyperplane for nonseparable patterns

- Consider now the more difficult and realistic case of nonseparable pattern (data) vectors in Section 6.3 in Haykin's book.
- We try to find an optimal hyperplane that minimizes the probability of classification error, averaged over the training set.
- The margin of separation between the classes is said to be **soft** if the constraint

$$d_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \tag{14}$$

cannot be satisfied for all data points $(\mathbf{x}_i, d_i)$, $i = 1, 2, \ldots, N$.

- This violation can arise in two ways, as illustrated in Figures 3 and 4.
- Nonseparable data points can be handled by introducing nonnegative **slack variables** ξ_i:

$$d_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \qquad i = 1, 2, \ldots, N \tag{15}$$

into the definition of the separating hyperplane (decision surface).

- For $0 \leq \xi_i \leq 1$ we have the case of Figure 3.
- If $\xi_i > 1$ we have the case of Figure 4 leading to a classification error.'
- In finding the optimal hyperplane, we try to keep the slack variables ξ_i as small as possible.
- The optimization procedure is fairly similar as in the case of linearly separable patterns in Section 6.2 in Haykin's book.
- The cost function is now

$$\begin{aligned} J(\mathbf{w}, b, \xi, \alpha, \mu) &= \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\mu_i\xi_i \\ &- \sum_{i=1}^{N}\alpha_i[d_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 + \xi_i] \end{aligned} \tag{16}$$

- Here C is a user-defined coefficient that controls the tradeoff between complexity (capacity) and number of classification errors.
- And $\sum_i \xi_i$ is an upper bound on the number of classification errors.
- The μ_i are new Lagrange multipliers needed to take into account the nonnegativity constraints $\xi_i \geq 0$.
- Similarly as for linearly separable pattern (training) vectors, the cost function $J(\mathbf{w}, b, \xi, \alpha, \mu)$ is optimized in two stages by:
  - By first optimizing it with respect to $\mathbf{w}$, b, and ξ_i;
  - And then solving the resulting dual problem for the α_i Lagrange multipliers.
- The first stage is carried out in an exercise problem.
- It turns out that the dual problem remains almost the same (11) as for the separable case.

- Except that the positivity constraints $\alpha_i \geq 0$ are replaced by the more stringent constraints $0 \leq \alpha_i \leq C$.
- **Support vectors:** All vectors within the margin of classification or on the wrong side of it.
- For more details, see Section 6.3 in Haykin's book and the exercises.
- The parameter C can be determined either:
  - Experimentally via the standard use of training and test set;
  - Or analytically by using theoretical results on VC dimension and generalization performance (not discussed in our course).

Nonlinear Support Vector Machines

- The material for this part is presented in more detail in Section 6.4 in Haykin's book.
- Consider now more realistic pattern recognition (classification) problems where:
  - The "optimal" decision surface is in general nonlinear, not a hyperplane;
  - The pattern (data) vectors are in general nonseparable, with some inevitable classification error.
- The basic mathematical framework is as follows:
  - Map the data vectors $\mathbf{x}$ into a high-dimensional **feature space:** $\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x})$
  - Construct an optimal separating hyperplane **in the feature space.**

- **Cover's theorem on separability of patterns** provides the justification for the first operation.
- It says that after a transformation into a high-dimensional feature space nonlinearly separated pattern vectors are more likely to be linearly separable there.
- The transformation must be nonlinear and the dimensionality of the feature space must be high enough for the Cover's theorem to hold.
- The optimal linear hyperplane is constructed in the feature space according to the theory just discussed in Section 6.3.
- The "kernel trick" is used to compute the corresponding nonlinear decision surface in the original smaller dimensional input space.
- The method is still computationally efficient since only the support vectors are utilized.

The "kernel trick"

- Let $\mathbf{x}$ be a m_0-dimensional input vector.
- Let $\varphi_1(\mathbf{x}), \varphi_2(\mathbf{x}), \ldots, \varphi_{m_1}(\mathbf{x})$ denote a set of nonlinear transforms from the input space to the m_1-dimensional feature space.
- The transformation functions $\varphi_j(\mathbf{x})$ are defined beforehand for all j.
- The equation

$$\sum_{j=0}^{m_1} w_j \varphi_j(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}) = 0 \tag{17}$$

  defines a hyperplane acting as a decision surface in the feature space.
- There the function $\varphi_0(\mathbf{x}) = 1$ corresponds to the bias term $w_0 = b$.
- The weight vector $\mathbf{w} = [w_0, w_1, \ldots, w_{m_1}]^T$, and the feature vector

$$\boldsymbol{\varphi}(\mathbf{x}) = [\varphi_0(\mathbf{x}), \varphi_1(\mathbf{x}), \ldots, \varphi_{m_1}(\mathbf{x})]^T \tag{18}$$

- The optimal linear separating hyperplane is found in the high-dimensional feature space using the methods discussed earlier.
- The solution is

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i d_i \boldsymbol{\varphi}(\mathbf{x}_i) \tag{19}$$

- The decision surface is

$$\sum_{i=1}^{N} \alpha_i d_i \boldsymbol{\varphi}^T(\mathbf{x}_i) \boldsymbol{\varphi}(\mathbf{x}) = 0 \tag{20}$$

- Define an **inner-product kernel** for $i = 1, 2, \ldots, N$ by

$$K(\mathbf{x}, \mathbf{x}_i) = \boldsymbol{\varphi}^T(\mathbf{x}) \boldsymbol{\varphi}(\mathbf{x}_i) = \sum_{j=0}^{m_1} \varphi_j(\mathbf{x}) \varphi_j(\mathbf{x}_i) \tag{21}$$

- If there exists a suitable simple form for the kernel K, **we need not compute the inner products in the high-dimensional feature**

space!

- The decision function is then

$$\sum_{i=1}^{N} \alpha_i d_i K(\mathbf{x}, \mathbf{x}_i) = 0 \tag{22}$$

Examples of Support Vector Machine

- The kernel $K(\mathbf{x}, \mathbf{x}_i)$ needs to satisfy Mercer's theorem but otherwise it can be arbitrary.
- Formal conditions of the Mercer's theorem in functional analysis have been presented on pp. 331-332 in Haykin's book.
- We skip that highly theoretical discussion.
- Intuitively, Mercer's theorem says that the kernel $K(\mathbf{x}, \mathbf{x}_i)$ must be positive definite.

- In the following, we summarize the inner-product kernels for three common types of support vector machines.
  1. **Polynomial learning machine.** The kernel is

$$K(\mathbf{x}, \mathbf{x}_i) = (\mathbf{x}^T \mathbf{x}_i + 1)^p \tag{23}$$

     where the power p is specified beforehand by the user.
  2. **Radial basis function network.** The kernel is

$$K(\mathbf{x}, \mathbf{x}_i) = \exp(-\frac{1}{2\sigma^2} \parallel \mathbf{x} - \mathbf{x}_i \parallel^2) \tag{24}$$

     The common width σ^2 of all kernels is specified by the user.
  3. **Two-layer perceptron.** The kernel is

$$K(\mathbf{x}, \mathbf{x}_i) = \tanh(\beta_0 \mathbf{x}^T \mathbf{x}_i + \beta_1) \tag{25}$$

     Mercer's theorem is satisfied only for some values of the scalar parameters β_0 and β_1.

Design of the SVM

- The expansion of the inner-product kernel $K(\mathbf{x}, \mathbf{x}_i)$ in Eq. (21) allows us to construct a decision surface that is:
  - Nonlinear in the input space;
  - But **its image in the feature space is linear.**
- The dual form of the optimization problem is still essentially the same, only the inner products have been changed into the kernels!
- One must find the Lagrange multipliers $\alpha_1, \alpha_2, \ldots, \alpha_N$ that maximize the objective function

$$Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j d_i d_j K(\mathbf{x}_i, \mathbf{x}_j) \qquad (26)$$

subject to the constraints

1. $\sum_{i=1}^{N} \alpha_i d_i = 0$

2. $0 \leq \alpha_i \leq C$ for $i = 1, 2, \ldots, N$ where C is a user-specified parameter.

- And again support vectors are the vectors for which $\alpha_i \neq 0$.
- The optimum value $\mathbf{w}_o$ of the linear weight vector in the feature space is

$$\mathbf{w}_o = \sum_{i=1}^{N} \alpha_{o,i} d_i \boldsymbol{\varphi}(\mathbf{x}_i) \qquad (27)$$

- Here $\alpha_{o,i}$ are the optimal values of the Lagrange multipliers.
- Note that the first component of $\mathbf{w}_o$ represents the optimum bias b_o.

Support vector machine as a feedforward network

- SVM is in effect a feedforward network in which:
  - Each hidden unit corresponds to one support vector.

  - The transfer function corresponds to the kernel K.
- The architecture of SVM network is shown in Figure 5.
- **The number of hidden units is chosen automatically to maximize separability,** given the type of kernel function.
- Usually, the model capacity (VC dimension) is controlled by varying the number of hidden units.
- In SVMs, the best possible solution in the high-dimensional (virtual) feature space is sought, which results in a choice of hidden units.
  - The feature space is high-dimensional to enhance separability.
  - Only a small subset of the vectors is chosen as support vectors.
  - By using the inner-product kernel the optimization can be done in the original low-dimensional space.

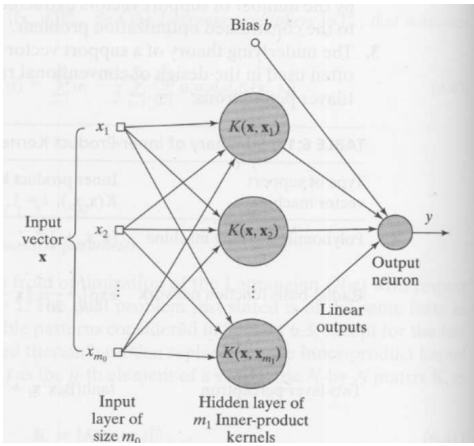

Figure 5: Architecture of support vector machine.

Simple example: XOR problem

- Consider the well-known XOR (Exclusive OR) problem.
- It is the simplest example of a two-class classification problem where the pattern vectors are not linearly separable.
- In the XOR problem, there are four two-dimensional pattern (input) vectors $\mathbf{x} = [x_1, x_2]^T$: $\mathbf{x}_1 = (-1, -1)$, $\mathbf{x}_2 = (-1, +1)$, $\mathbf{x}_3 = (+1, -1)$, and $\mathbf{x}_4 = (+1, +1)$.
- $\mathbf{x}_1$ and $\mathbf{x}_4$ belong to the class 1 with the desired response $d_i = -1$, while $\mathbf{x}_2$ and $\mathbf{x}_3$ belong to the class 2 with $d_i = +1$.
- Let us choose the second-order polynomial kernel

$$\begin{aligned} K(\mathbf{x}, \mathbf{x}_i) &= (1 + \mathbf{x}^T \mathbf{x}_i)^2 \\ &= 1 + x_1^2 x_{i1}^2 + 2x_1 x_2 x_{i1} x_{i2} + x_2^2 x_{i2}^2 + 2x_1 x_{i1} + 2x_2 x_{i2} \end{aligned} \quad (28)$$

- The kernel corresponds to the inner product of feature vectors

$$K(\mathbf{x}, \mathbf{x}_i) = \boldsymbol{\varphi}^T(\mathbf{x}) \boldsymbol{\varphi}(\mathbf{x}_i) \quad (29)$$

where

$$\boldsymbol{\varphi}(\mathbf{x}) = [1, x_1^2, \sqrt{2} x_1 x_2, x_2^2, \sqrt{2} x_1, \sqrt{2} x_2]^T \quad (30)$$

- It is easy to compute the values $K(\mathbf{x}_i, \mathbf{x}_j)$ for the four pattern vectors x_k and evaluate the objective function (26).
- The details of the optimization procedure can be found in Section 6.5 in Haykin's book.
- It turns out that in this simple example all the four input vectors $\mathbf{x}_i$ are support vectors.
- The optimum values of the Lagrange multipliers are the same $\alpha_{o,i} = 1/8$ for all of them.

- The optimum value of the weight vector is

$$\begin{aligned} \mathbf{w}_o &= \frac{1}{8}[-\phi(\mathbf{x}_1) + \phi(\mathbf{x}_2) + \phi(\mathbf{x}_3) - \phi(\mathbf{x}_4)] \\ &= [0, 0, -1/\sqrt{2}, 0, 0, 0]^T \end{aligned} \quad (31)$$

- The optimal hyperplane in the feature space

$$\mathbf{w}_o^T \boldsymbol{\varphi}(\mathbf{x}) = 0 \quad (32)$$

boils in the input space down to the simple nonlinear equation

$$-x_1 x_2 = 0 \quad (33)$$

- Figure 6 (a) shows a diagram of the resulting simple polynomial SVM, and Figure 6 (b) shows that it can classify all the 4 input vectors $\mathbf{x}_i$ correctly.
- In Fig. 6 (b) the second vector mapping to 1.0 should be $(-1, -1)$.

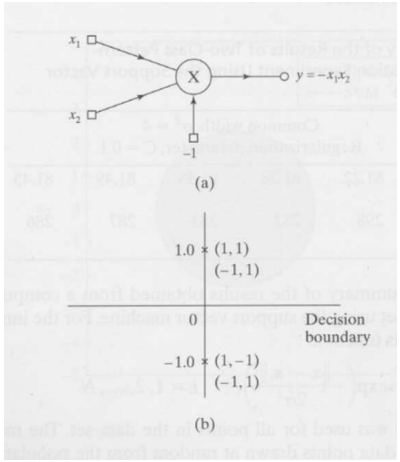

Figure 6: (a) Polynomial machine for solving the XOR problem. (b) Images of the four data point of the XOR problem in the feature space

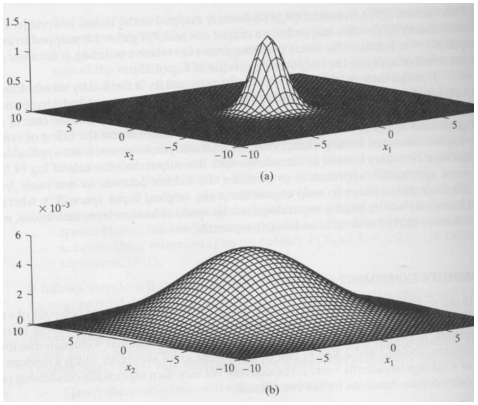

Figure 7: The Gaussian probability density functions of class $\mathcal{C}_1$ (upper figure a) and $\mathcal{C}_2$ (lower figure b).

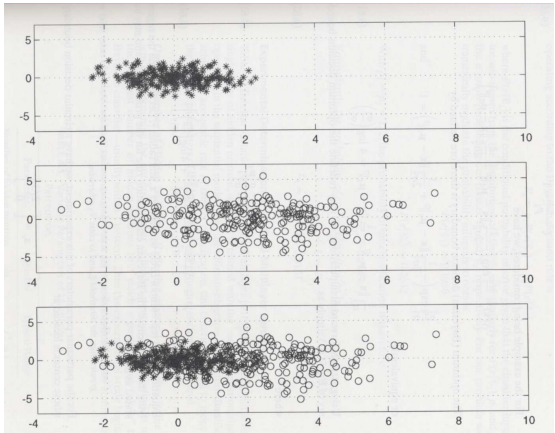

Figure 8: Data vectors of class $\mathcal{C}_1$ (top), class $\mathcal{C}_2$ (middle), and both the classes (bottom).

Computer experiment: two overlapping Gaussians

The problem and using MLP networks for it

- Recall the classification problem discussed in more detail at the end of lecture 4.
- The two-dimensional data vectors belong to two overlapping classes $\mathcal{C}_1$ and $\mathcal{C}_2$.
- The Gaussian probability densities of the classes are reproduced in Figure 7.
- Figure 8 shows samples of the data vectors.
- Due to the overlap, the theoretically optimal Bayes classifier achieves the classification rate 81.51%.
- The optimum decision boundary is a circle of center $\mathbf{x}_c = [-2/3, 0]^T$ and radius $r = 2.34$.

- In lecture 4 (Section 4.8 in Haykin's book), MLP networks trained by the backpropagation algorithm were tested in this problem.
- In turned out that an MLP network with one hidden layer of two neurons only provided sufficient performance.
- Which can be improved only slightly by using more hidden neurons and training data.
- The average classification accuracy for 20 independent such MLP networks trained using 1000 sample vectors was 79.40%.
- For your convenience, the decision boundaries given by 3 best performing MLP networks and 3 worst ones are reproduced in Figures 9 and 10.

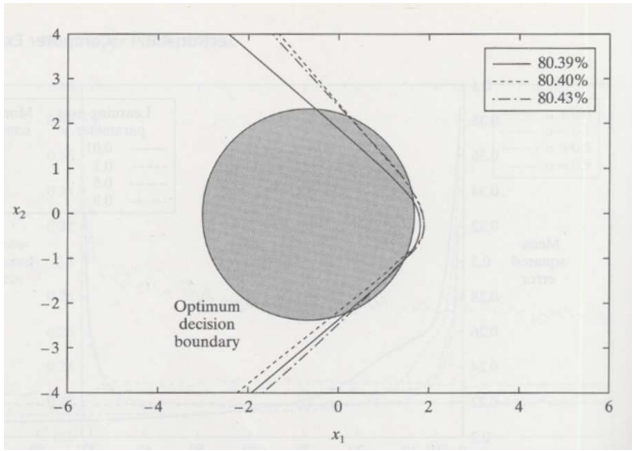

Figure 9: Three best decision boundaries given be the MLP network trained using the backpropagation algorithm.

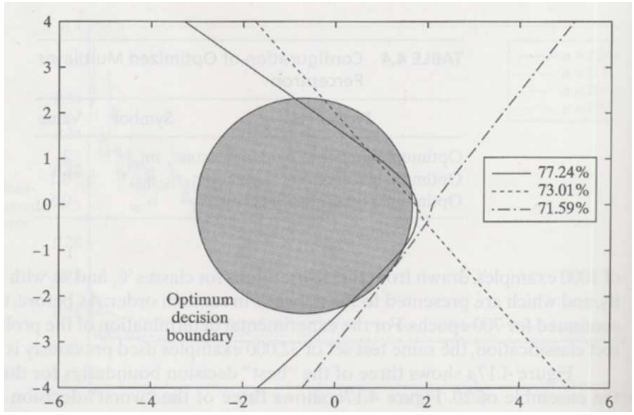

Figure 10: Three worst decision boundaries given be the MLP network trained using the backpropagation algorithm.

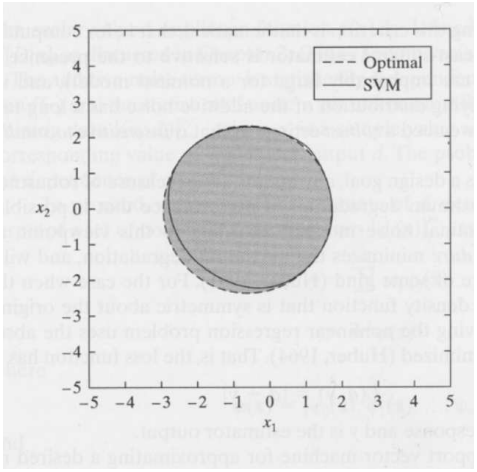

Figure 11: Decision boundary given by the support vector machine.

Results for the support vector machine

- In Haykin's Section 6.6, support vector machines are tested in the same problem.
- The inner-product kernel used was the radial basis function

$$K(\mathbf{x}, \mathbf{x}_i) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{2\sigma^2}\right) \tag{34}$$

- The common width of RBF kernels was $\sigma^2 = 4$, and the regularization parameter was $C = 0.1$.
- For training, 500 randomly chosen data vectors were used.
- In five experiments with different training sets, the average classification rate of 81.40% was achieved.
- It is very close to the optimum of 81.51%.
- One of the almost perfect decision boundaries given by SVM is

shown in Figure 11.

- The price paid for the excellent performance of SVM is that it uses nearly 300 support vectors (60% of the training sample)!
- While the MLP network used only two hidden neurons.

Classification of real-world MNIST data

- The MNIST data set consists of digital 28×28 images of handwritten digits $0, 1, \ldots, 9$.
- It is a widely used benchmark data set for testing the classification performance of various machine learning methods.
- The data and its description are available at `http://yann.lecun.com/exdb/mnist/index.html`
- The database contains 60,000 training and 10,000 test images.

- In most methods, the images are scanned row-by-row or column-by-column to $28 \times 28 = 784$-dimensional vectors.
- The best error rate achieved using SVMs is 1.4%.
- While for the MLP networks the error rate is 1.6%.

A comparison of SVM and MLP networks

- In general, support vector machines and MLP networks have the following **pros and cons**.
- **Support vector machines:**
  - \+ Ability to achieve a close to optimum performance in classification problems.
  - \+ No problem specific prior knowledge is needed for that.
  - – Computationally complex, requiring lots of support vectors for achieving best performance.

- **Multilayer perceptron networks:**
  - \+ Provide a computationally efficient solution.
  - – Require tuning of a multitude of design parameters.
  - – Long learning times and problems with local minima for large problems.

SVMs for nonlinear regression

The nonlinear regression problem

- The material for this part is presented in more detail in Sections 6.7 and 6.8 in Haykin's book.
- Thus far we have considered the use of support vector machines for **classification** (pattern recognition) problems.
- There the goal is to build a class border (decision boundary) of the

form

$$\mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}) = 0 \tag{35}$$

- This is a hyperplane in the high-dimensional feature space, but a nonlinear surface in the lower-dimensional input space.
- Consider now a **nonlinear regressive model** where a scalar variable d depends nonlinearly on a vector $\mathbf{x}$ via

$$d = f(\mathbf{x}) + \nu \tag{36}$$

- Here f is an unknown function and ν is statistically independent noise term which is also unknown.
- Again, we have only a training set $\{(\mathbf{x}_i, d_i)\}_{i=1}^N$ available.
- In **regression** with SVMs the goal is to approximate d with the

output variable y computed from

$$y = \mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}) = \sum_{j=0}^{m_1} w_j \varphi_j(\mathbf{x}) \tag{37}$$

where the vectors $\boldsymbol{\varphi}(\mathbf{x})$ and $\mathbf{w}$ are defined by

$$\boldsymbol{\varphi}(\mathbf{x}) = [\varphi_0(\mathbf{x}), \varphi_1(\mathbf{x}), \ldots, \varphi_{m_1}(\mathbf{x})]^T \tag{38}$$

$$\mathbf{w} = [w_0, w_1, \ldots, w_{m_1}]^T \tag{39}$$

- As before, $\varphi_0(\mathbf{x}) = 1$, so that the weight w_0 represents the bias b.
- Note that now there will (almost) always be errors in the output, and all of the erroneous input vectors would become support vectors!
- This problem can be handled by introducing a robust ϵ-**insensitive loss function.**

ϵ-insensitive loss function

- Quadratic error criteria, such as the mean-square error or least-squares error, are used routinely in many machine learning methods.
- For example in MLP and RBF networks, principal component analysis, LMS algorithm and so on.
- The reason for their popularity is mathematical convenience.
- Optimization of quadratic criteria often leads to linear equations that can be solved in closed form.
- However, quadratic error criteria are sensitive to large errors.
- To see this, let us for simplicity consider a scalar error e.
- Compare now the quadratic error criterion $L_2(e) = e^2$ and absolute error criterion $L_1(e) =\mid e \mid$.

- For the error (scaled suitably) $e = 1$, both $L_1(e) = L_2(e) = 1$.
- However, if $e = 10$, $L_1(e) = 10$, but $L_2(e) = 100$.
- Thus quadratic error criteria weight heavily large errors.
- The optimum of quadratic criteria may depend almost completely on a few large values of the error e.
- While small values of error e may have a negligible effect on the optimum.
- Large errors are often due to **outliers** in the data originating from measurement etc. errors.
- The absolute error criterion $L_1(e) =\mid e \mid$ is much more **robust** against large values of the error.
- Therefore it would be preferable in many cases.
- But absolute error is clearly harder to optimize.

- In our case, the error $e = d - y$.
- Consider a modification of the plain absolute error criterion

$$L_\epsilon(e) = \begin{cases} |e| - \epsilon \text{ for } |e| \geq \epsilon \\ 0 \text{ for } |e| \leq \epsilon \end{cases} \tag{40}$$

- The criterion (40) is called ϵ-**insensitive loss function.**
- It is otherwise like standard absolute error, but there is a zero error zone $[-\epsilon, +\epsilon]$ around the origin.
- See Figure 12.
- The loss function (40) is robust, but furthermore also insensitive to small changes is the regressive model.

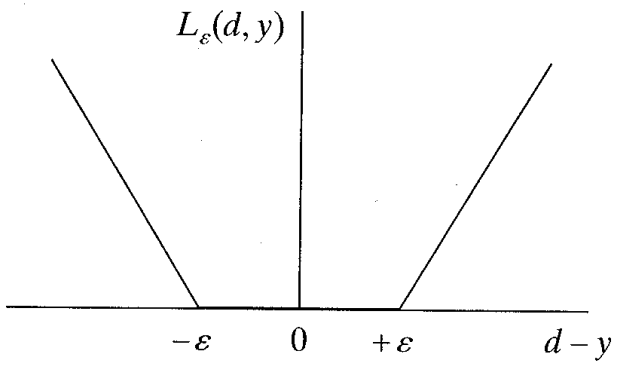

Figure 12: ϵ-insensitive loss function.

The SVM cost function and solution for nonlinear regression

- The total cost function to be minimized is the empirical risk

$$R_{emp} = \frac{1}{N}\sum_{i=1}^{N} L_\epsilon(e_i) = \frac{1}{N}\sum_{i=1}^{N} L_\epsilon(d_i, y_i) \tag{41}$$

  subject to the inequality

$$\| \mathbf{w} \|^2 \leq c_0 \tag{42}$$

- There c_0 is a constant and $L_\epsilon(d_i, y_i)$ the ϵ-insensitive loss function.
- This constrained optimization problem can be reformulated by introducing two sets of nonnegative **slack variables** ξ_i and ξ_i' as follows:

$$d_i - \mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}_i) \leq \epsilon + \xi_i \tag{43}$$

$$\mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}_i) - d_i \leq \epsilon + \xi_i' \tag{44}$$

$$\xi_i \geq 0, \qquad i = 1, 2, \ldots, N \tag{45}$$

$$\xi_i' \geq 0, \qquad i = 1, 2, \ldots, N \tag{46}$$

- The slack variables ξ_i and ξ_i', $i = 1, 2, \ldots, N$, measure the distance of y from the "ϵ-tube".
- The cost function to be minimized is then a sum of the distances from the tube, and the complexity cost:

$$\Phi(\mathbf{x}, \xi, \xi') = C\sum_{i=1}^{N}(\xi + \xi') + \frac{1}{2}\|\mathbf{w}\|^2 \tag{47}$$

- The term $\frac{1}{2}\|\mathbf{w}\|^2$ replaces the inequality (42).
- The optimization of this constrained cost function is somewhat more complicated than previously.
- We skip the details of derivation on page 342.

- The result is that the nonlinear regression problem for SVMs can be cast in the form of the **dual problem:**
- Maximize

$$Q(\alpha, \alpha') = \sum_{i=1}^{N} d_i(\alpha_i - \alpha_i') - \epsilon\sum_{i=1}^{N}(\alpha_i + \alpha_i') - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i - \alpha_i')(\alpha_j - \alpha_j')K(\mathbf{x}_i, \mathbf{x}_j) \tag{48}$$

  with respect to the Lagrange multipliers $\{\alpha_i\}_{i=1}^N$ and $\{\alpha_i'\}_{i=1}^N$, subject to the constraints

  1. $\sum_{i=1}^N(\alpha_i - \alpha_i') = 0$
  2. $0 \leq \alpha_i \leq C, \quad 0 \leq \alpha_i' \leq C$, $i = 1, 2, \ldots, N$

- The solution is

$$y = \sum_{i=1}^{N} (\alpha_i - \alpha_i')K(\mathbf{x}, \mathbf{x}_i) \tag{49}$$

- The **support vectors** are the vectors outside the "tube", i.e., the vectors for which $\alpha_i \neq \alpha_i'$.
- Note that now there are two free parameters:
  - C that controls the tradeoff between complexity (capacity) and the amount (and size) of errors
  - ϵ that controls the sensitivity of the loss function to small errors
- These parameters must be tuned simultaneously.
- Regression is also intrinsically more difficult than classification.
- SVM for nonlinear regression may be implemented using a polynomial learning machine, radial basis function network, or two-layer perceptron as discussed before.

Extensions and modifications

- In Du's and Swamy's book "Neural Networks and Statistical Learning", Springer 2014, many extensions and modifications of SVMs are presented in Chapter 16.
- However, their discussion is often superficial with no mathematics.
- An important modification is least-squares SVM in Section 16.4.
- It simplifies greatly the training process of SVM by replacing the inequality constraints with equality ones.
- Various training methods for SVM's are discussed in Section 16.5.
- In the remainder of Chapter 16, pruning SVMs, multiclass SVMs, and support vector clustering are discussed.
- As well as some other modifications in Sections 16.10 - 16.14.

Summary and discussion

- The support vector machine is an elegant and highly principled learning method.
- For designing a feedforward network with a single layer of hidden units.
- It provides a fairly straightforward engineering solution for classification tasks.
- Model complexity can be controlled independently of the dimensionality.
- It is (relatively) easy to solve the dual problem:
  - Quadratic problem $\Rightarrow$ a global extremum will be found.
  - The computation can be made in an effective fashion (but for very large data sets special methods may be needed).

- However, SVMs are slower than for example BP algorithms.
- And SVM operates in batch mode only.
- The error criteria used in SVMs are usually more appropriate than the least-squares error used in MLP and RBF networks:
  - In pattern classification problems, SVM minimizes the number of training samples that fall inside the margin of separation.
  - In nonlinear regression, SVMs minimize the robust ϵ-insensitive absolute error criterion.
- Key ideas of support vector machine in brief:
  - Constructs a separating hyperplane in a high-dimensional feature space: $\mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x})$.
  - Maximizes separability: with the aid of the slack variables ξ_i.
  - Expresses the hyperplane in the original space using a small set of training vectors, the "support vectors":

$\mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i d_i K(\mathbf{x}, \mathbf{x}_i)$.

  - Is nonlinear in the input space.

- Yet another universal feedforward network, but computes automatically the number of hidden units.
- But note that the number of hidden units is typically a fraction of the number of training samples, which needs to be selected.
- Suitable tasks: pattern classification, nonlinear regression.
- Has a connection to structural risk minimization (VC dimension).