Introduction to Artificial Intelligence

George Voutsadakis[1]

[1]Mathematics and Computer Science
Lake Superior State University

LSSU Math 400

Introduction

- Support Vector Machine (SVM) constitutes a special type of kernel-based algorithm that has sparse solutions.
- As a result, predictions for new inputs depend only on the kernel function evaluated at a subset of the training data points.
- SVM became popular for solving problems in classification, regression, and novelty detection.
- An important property is that the determination of the model parameters corresponds to a convex optimization problem, and so any local solution is also a global optimum.
- The discussion of support vector machines makes extensive use of Lagrange multipliers.
- The SVM is a decision machine and so does not provide posterior probabilities.

Subsection 1

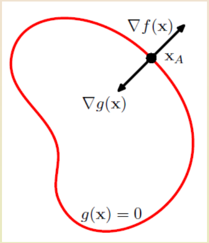
Review of Lagrange Multipliers

Without Lagrange Multipliers

- **Lagrange multipliers** are used to find the extreme points of a function of several variables subject to one or more constraints.
- Consider the problem of finding the maximum of a function $f(x_1, x_2)$ subject to a constraint relating x_1 and x_2, which we write in the form $g(x_1, x_2) = 0$.
- One approach would be to
  - solve the constraint, thus expressing x_2 as a function of x_1: $x_2 = h(x_1)$;
  - substitute into $f(x_1, x_2)$ to give a function of x_1: $f(x_1, h(x_1))$;
  - find the maximum with respect to x_1 by differentiation in the usual way.
- This approach may entail drawbacks:
  - It may be difficult to find an analytic solution of the constraint equation that allows x_2 to be expressed as an explicit function of x_1.
  - Variables x_1 and x_2 are treated differently, whence the natural symmetry between these variables is spoiled.

∇g is Normal to $g = 0$

- Consider a D-dimensional variable $\mathbf{x}$ with components $x_1, \ldots, x_D$.

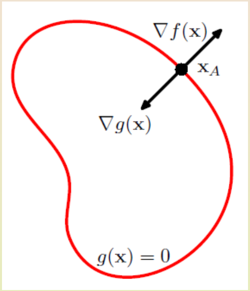


- The constraint equation $g(\mathbf{x}) = 0$ represents a $(D-1)$-dimensional surface in $\mathbf{x}$-space:
- At any point on the constraint surface the gradient $\nabla g(\mathbf{x})$ of the constraint function will be orthogonal to the surface:
- If $\mathbf{x}$ lies on the constraint surface, consider $\mathbf{x} + \epsilon$ that also lies on the surface.

Take a Taylor expansion around $\mathbf{x}$: $g(\mathbf{x} + \epsilon) \simeq g(\mathbf{x}) + \epsilon^T \nabla g(\mathbf{x})$. Because both $\mathbf{x}$ and $\mathbf{x} + \epsilon$ lie on the constraint surface, we have $g(\mathbf{x}) = g(\mathbf{x} + \epsilon)$ and hence $\epsilon^T \nabla g(\mathbf{x}) \simeq 0$. In the limit $\|\epsilon\| \to 0$ we have $\epsilon^T \nabla g(\mathbf{x}) = 0$. Since ϵ is then parallel to the constraint surface $g(\mathbf{x}) = 0$, we see that the vector ∇g is normal to the surface.

Lagrange Multipliers

- We seek a point $\mathbf{x}^*$ on the constraint surface such that $f(\mathbf{x})$ is maximized.



- Such a point must have the property that the vector $\boldsymbol{\nabla} f(\mathbf{x})$ is also orthogonal to the constraint surface: Otherwise we could increase the value of $f(\mathbf{x})$ by moving a short distance along the constraint surface.
- Thus $\boldsymbol{\nabla} f$ and $\boldsymbol{\nabla} g$ are parallel (or anti-parallel) vectors, and so there must exist a parameter λ such that $\boldsymbol{\nabla} f + \lambda \boldsymbol{\nabla} g = \mathbf{0}$, where $\lambda \neq 0$ is known as a **Lagrange multiplier**.
- Note that λ can have either sign.

Optimizing the Lagrangian

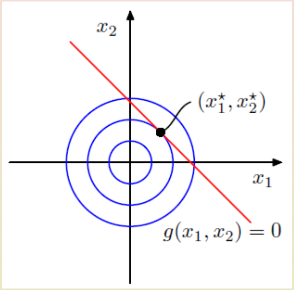
- The **Lagrangian function** is defined by

$$L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda g(\mathbf{x}).$$

- The previous condition is obtained by setting $\nabla_{\mathbf{x}} L = \mathbf{0}$.
- Furthermore, the condition $\frac{\partial L}{\partial \lambda} = 0$ yields $g(\mathbf{x}) = 0$.
- Thus, to find the maximum of a function $f(\mathbf{x})$ subject to the constraint $g(\mathbf{x}) = 0$:
    - Form the Lagrangian function;
    - Find the extreme point of $L(\mathbf{x}, \lambda)$ with respect to both $\mathbf{x}$ and λ.
- For a D-dimensional vector $\mathbf{x}$, this gives $D + 1$ equations that determine both the extreme point $\mathbf{x}^*$ and the value of λ.
- If we are only interested in $\mathbf{x}^*$, we can eliminate λ.

An Example Illustrating the Method

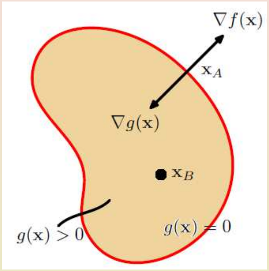
- Suppose we wish to find the stationary point of the function
$$f(x_1, x_2) = 1 - x_1^2 - x_2^2$$
subject to the constraint
$g(x_1, x_2) = x_1 + x_2 - 1 = 0$



- The corresponding Lagrangian function is given by
$$L(\mathbf{x}, \lambda) = 1 - x_1^2 - x_2^2 + \lambda(x_1 + x_2 - 1).$$
- The conditions for this Lagrangian to be stationary with respect to x_1, x_2 and λ give
$$\begin{array}{rcl} -2x_1 + \lambda & = & 0 \\ -2x_2 + \lambda & = & 0 \\ x_1 + x_2 - 1 & = & 0 \end{array}$$
- Solving, we get $(x_1^*, x_2^*) = (\frac{1}{2}, \frac{1}{2})$, and $\lambda = 1$.

Optimization Subject to Inequality Constraints

- We consider: Maximize $f(\mathbf{x})$ subject to an inequality constraint of the form $g(\mathbf{x}) \geq 0$.
- There are now two kinds of solution possible
  - The constrained stationary point may lie in the region where $g(\mathbf{x}) > 0$; In which case the constraint is **inactive**.
  - It may also lie on the boundary $g(\mathbf{x}) = 0$; In this case the constraint is said to be **active**.



- In the former case, the function $g(\mathbf{x})$ plays no role and the extremal condition is $\boldsymbol{\nabla} f(\mathbf{x}) = \mathbf{0}$, corresponding to $\lambda = 0$.
- In the case where the solution lies on the boundary, we get an extreme point of the Lagrange function, with $\lambda \neq 0$.
- Now, however, the sign of the Lagrange multiplier is crucial, because the function $f(\mathbf{x})$ will only be at a maximum if its gradient is oriented away from the region $g(\mathbf{x}) > 0$.
- So $\boldsymbol{\nabla} f(\mathbf{x}) = -\lambda \boldsymbol{\nabla} g(\mathbf{x})$, for some value of $\lambda > 0$.

Karush-Kuhn-Tucker Conditions

- For either case, the product $\lambda g(\mathbf{x}) = 0$.
- Thus, the solution to the problem of maximizing $f(\mathbf{x})$ subject to $g(\mathbf{x}) \geq 0$ is obtained by optimizing the Lagrange function

$$L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda g(\mathbf{x})$$

with respect to $\mathbf{x}$ and λ subject to the conditions

$$\begin{aligned} g(\mathbf{x}) &\geq 0 \\ \lambda &\geq 0 \\ \lambda g(\mathbf{x}) &= 0 \end{aligned}$$

- These are known as the **Karush-Kuhn-Tucker** (**KKT**) **conditions**.
- Note that if we wish to minimize (rather than maximize) the function $f(\mathbf{x})$ subject to an inequality constraint $g(\mathbf{x}) \geq 0$, then we minimize the Lagrangian function $L(\mathbf{x}, \lambda) = f(\mathbf{x}) - \lambda g(\mathbf{x})$ with respect to $\mathbf{x}$, again subject to $\lambda \geq 0$.

Multiple Equality and Inequality Constraints

- It is straightforward to extend the technique of Lagrange multipliers to the case of multiple equality and inequality constraints.
- Suppose we wish to maximize $f(\mathbf{x})$ subject to $g_j(\mathbf{x}) = 0$ for $j = 1, \ldots, J$, and $h_k(\mathbf{x}) \geq 0$, for $k = 1, \ldots, K$.
- We then introduce **Lagrange multipliers** $\{\lambda_j\}$ and $\{\mu_k\}$, and then optimize the **Lagrangian function** given by

$$L(\mathbf{x}, \{\lambda_j\}, \{\mu_k\}) = f(\mathbf{x}) + \sum_{j=1}^{J} \lambda_j g_j(\mathbf{x}) + \sum_{k=1}^{K} \mu_k h_k(\mathbf{x}),$$

subject to $\mu_k \geq 0$ and $\mu_k h_k(\mathbf{x}) = 0$ for $k = 1, \ldots, K$.

Subsection 2

Maximum Margin Classifiers

The Two-Class Classification Problem

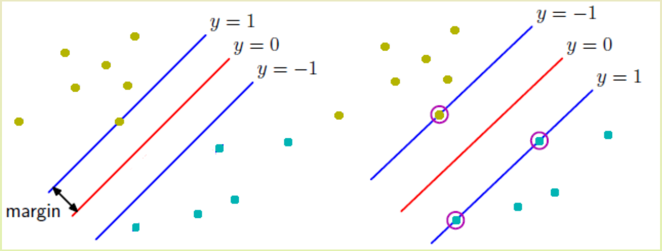
- We return to the two-class classification problem using linear models of the form $y(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x}) + b$, where $\phi(\mathbf{x})$ denotes a fixed feature-space transformation, and b is a bias parameter.
- The training data set comprises N input vectors $\mathbf{x}_1, \ldots, \mathbf{x}_N$, with corresponding target values $t_1, \ldots, t_N$ where $t_n \in \{-1, 1\}$, and new data points $\mathbf{x}$ are classified according to the sign of $y(\mathbf{x})$.
- Assume for the moment that the training data set is linearly separable in feature space, so that there exists at least one choice of the parameters $\mathbf{w}$ and b such that the function $y(\mathbf{x})$ satisfies

$$y(\mathbf{x}_n) > 0, \text{ if } t_n = +1, \quad \text{and} \quad y(\mathbf{x}_n) < 0, \text{ if } t_n = -1.$$

- Note this means that $t_n y(\mathbf{x}_n) > 0$ for all training data points.
- There may of course exist many such solutions.
- We studied the perceptron algorithm that is guaranteed to find a solution in a finite number of steps.

Margin

- The solution that the perceptron finds depends on
  - the initial values chosen for **w** and *b* and
  - the order in which the data points are presented.
- If there are multiple solutions, then we should try to find the one that will give the smallest generalization error.
- The support vector machine approaches this problem through the concept of the margin, which is defined to be the smallest distance between the decision boundary and any of the samples.



Maximum Margin

- In support vector machines the decision boundary is chosen to be the one for which the margin is maximized.
- The perpendicular distance of a point $\mathbf{x}$ from a hyperplane defined by $y(\mathbf{x}) = 0$ where $y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b$ is given by $\frac{|y(\mathbf{x})|}{\|\mathbf{w}\|}$.
- We are only interested in solutions for which all data points are correctly classified, so that $t_n y(\mathbf{x}_n) > 0$ for all n.
- Thus, the distance of a point $\mathbf{x}_n$ to the decision surface is given by $\frac{t_n y(\mathbf{x}_n)}{\|\mathbf{w}\|} = \frac{t_n(\mathbf{w}^T \phi(\mathbf{x}_n) + b)}{\|\mathbf{w}\|}$.
- The margin is given by the perpendicular distance to the closest point $\mathbf{x}_n$ from the data set and we wish to optimize the parameters $\mathbf{w}$ and b in order to maximize this distance.
- Thus the maximum margin solution is found by solving

$$\operatorname*{argmax}_{\mathbf{w},b} \left\{ \frac{1}{\|\mathbf{w}\|} \min_n \left[t_n (\mathbf{w}^T \phi(\mathbf{x}_n) + b) \right] \right\}.$$

The Canonical Representation of the Decision Hyperplane

- Direct solution of this optimization problem is very complex.
- We convert it into an easier to solve equivalent problem.
- If we re-scale $\mathbf{w} \to \kappa \mathbf{w}$ and $b \to \kappa b$, then the distance from any point $\mathbf{x}_n$ to the decision surface, given by $\frac{t_n y(\mathbf{x}_n)}{\|\mathbf{w}\|}$ is unchanged
- We can, thus, set $t_n(\mathbf{w}^T \phi(\mathbf{x}_n) + b) = 1$ for the point that is closest to the surface.
- Then, all data points will satisfy

$$t_n(\mathbf{w}^T \phi(\mathbf{x}_n) + b) \geq 1, \quad n = 1, \ldots, N.$$

- This is the **canonical representation of the decision hyperplane**.
- The constraints are said to be **active** for those data points for which equality holds, and **inactive** for the remainder.
- By definition, there will always be at least one active constraint, because there will always be a closest point.
- Moreover, once the margin has been maximized there will be at least two active constraints.

Reduction to Quadratic Programming Optimization

- The optimization problem now requires that we maximize $\frac{1}{\|\mathbf{w}\|}$.
- This is equivalent to minimizing $\|\mathbf{w}\|^2$.
- So we have to solve the optimization problem

$$\underset{\mathbf{w},b}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{w}\|^2,$$

  subject to the constraints

$$t_n(\mathbf{w}^T\phi(\mathbf{x}_n) + b) \geq 1, \quad n = 1, \ldots, N.$$

- The factor of $\frac{1}{2}$ is included for later convenience.
- This is an example of a quadratic programming problem in which we are trying to minimize a quadratic function subject to a set of linear inequality constraints.
- The bias parameter b is determined implicitly, because the constraints require that changes to $\|w\|$ be compensated by changes to b.

The Dual Representation

- To solve, introduce Lagrange multipliers $a_n \geq 0$, with one multiplier a_n for each of the constraints.
- With $\mathbf{a} = (a_1, \ldots, a_N)^T$, we get the Lagrangian function
$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{n=1}^{N} a_n[t_n(\mathbf{w}^T\phi(\mathbf{x}_n) + b) - 1].$$
- The minus in front of the Lagrange term is due to minimization with respect to $\mathbf{w}$ and b, and maximization with respect to $\mathbf{a}$.
- Calculate the derivatives of $L(\mathbf{w}, b, \mathbf{a})$ with respect to $\mathbf{w}$ and b and set them equal to zero:
$$\mathbf{w} = \textstyle\sum_{n=1}^N a_n t_n \phi(\mathbf{x}_n) \qquad 0 = \sum_{n=1}^N a_n t_n.$$
- Use these to eliminate $\mathbf{w}$ and b from $L(\mathbf{w}, b, \mathbf{a})$. Obtain the dual representation: maximize with respect to $\mathbf{a}$
$$\widetilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m), \; k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T\phi(\mathbf{x}')$$
subject to $a_n \geq 0$, $n = 1, \ldots, N$, and $\sum_{n=1}^N a_n t_n = 0$.

Advantages of Passing to the Dual

- The solution to a quadratic programming problem in M variables in general has computational complexity $\mathrm{O}\left(M^3\right)$.
- In going to the dual formulation we have turned the original optimization problem, which involved minimization over M variables, into the dual problem, which has N variables.
- For a fixed set of basis functions whose number M is smaller than the number N of data points, the move to the dual problem appears disadvantageous.
- However, it allows the model to be reformulated using kernels:
  - The maximum margin classifier can be applied efficiently to feature spaces whose dimensionality exceeds the number of data points, including infinite feature spaces.
  - The kernel formulation also makes clear the role of the constraint that the kernel function $k(\mathbf{x}, \mathbf{x}')$ be positive definite; This ensures that the Lagrangian function $\widetilde{L}(\mathbf{a})$ is bounded below; In turn, this gives rise to a well defined optimization problem.

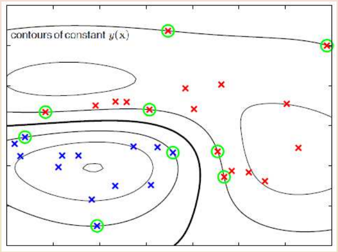
Classification of New Data Points

- To classify a new $\mathbf{x}$, we evaluate $y(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x}) + b$.
- In terms of the parameters $\{a_n\}$ and the kernel function, we get $y(\mathbf{x}) = \sum_{n=1}^N a_n t_n k(\mathbf{x}, \mathbf{x}_n) + b$.
- It can be shown that a constrained optimization of this form satisfies the Karush-Kuhn-Tucker (KKT) conditions:
$$a_n \geq 0, \quad t_n y(\mathbf{x}_n) - 1 \geq 0, \quad a_n[t_n y(\mathbf{x}_n) - 1] = 0.$$
- Thus for every data point, either $a_n = 0$ or $t_n y(\mathbf{x}_n) = 1$.
- Any data point for which $a_n = 0$ will not appear in the sum and hence plays no role in making predictions for new data points.
- The remaining data points are called **support vectors**, and because they satisfy $t_n y(\mathbf{x}_n) = 1$, they correspond to points that lie on the maximum margin hyperplanes in feature space.
- This property is central to the practical applicability of support vector machines. Once the model is trained, a significant proportion of the data points can be discarded and only the support vectors retained.

Value of the Threshold Parameter b

- Having solved the quadratic programming problem for **a**, we can then determine the value of b by noting that any support vector $\mathbf{x}_n$ satisfies $t_n y(\mathbf{x}_n) = 1$.
- This gives
$$t_n\left(\sum_{m \in \mathcal{S}} a_m t_m k(\mathbf{x}_n, \mathbf{x}_m) + b\right) = 1,$$
where $\mathcal{S}$ is the set of indices of the support vectors.
- Multiplying through by t_n, making use of $t_n^2 = 1$, and then averaging over all support vectors and solving for b, gives
$$b = \frac{1}{N_{\mathcal{S}}} \sum_{n \in \mathcal{S}} \left(t_n - \sum_{m \in \mathcal{S}} a_m t_m k(\mathbf{x}_n, \mathbf{x}_m) \right),$$
where $N_{\mathcal{S}}$ is the total number of support vectors.

Illustration of an SVM Classification



- The figure shows an example of the classification resulting from training a support vector machine on a simple data set using a Gaussian kernel

$$k(\mathbf{x},\mathbf{x}') = e^{-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma^2}}.$$

- The data set is not linearly separable in the data space **x**.
- It is linearly separable in the nonlinear feature space defined implicitly by the nonlinear kernel function.
- Thus, training data are perfectly separated in the original space.
- We get a geometrical insight into the origin of sparsity in the SVM. The maximum margin hyperplane is defined by the location of the support vectors; other data points can be moved around freely without changing the decision boundary.

Subsection 3

Overlapping Class Distributions

Handling Non-Separability of Training Points

- Thus far, we have assumed that the training data points are linearly separable in the feature space $\phi(\mathbf{x})$.
- The resulting support vector machine will give exact separation of the training data in the original input space $\mathbf{x}$, although the corresponding decision boundary will be nonlinear.
- In practice the class-conditional distributions may overlap, in which case exact separation can lead to poor generalization.
- We need a way to modify the support vector machine so as to allow some of the training points to be misclassified.
- In the case of separable classes, an error function that gave infinite error if a data point was misclassified and zero error if it was classified correctly was used.
- We modify this approach so that data points are allowed to be on the "wrong side" of the margin boundary, but with a penalty that increases with the distance from that boundary.

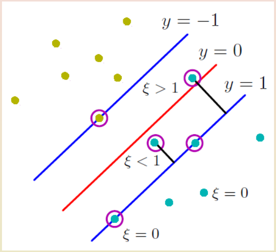
Introducing Slack Error Variables

- It is convenient to make this penalty a linear function of the distance.
- We introduce slack variables, $\xi_n \geq 0$, $n = 1, \ldots, N$, with one slack variable for each training data point.
- These are defined by $\xi_n = 0$ for data points that are on or inside the correct margin boundary and $\xi_n = |t_n - y(\mathbf{x}_n)|$ for other points.
- A data point that is on the decision boundary $y(\mathbf{x}_n) = 0$ will have $\xi_n = 1$, and points with $\xi_n > 1$ will be misclassified.
- The exact classification constraints are replaced by

$$t_n y(\mathbf{x}_n) \geq 1 - \xi_n, \quad n = 1, \ldots, N,$$

  in which the slack variables are constrained to satisfy $\xi_n \geq 0$.
  - Data points for which $\xi_n = 0$ are correctly classified and are either on the margin or on the correct side of the margin.
  - Points for which $0 < \xi_n \leq 1$ lie inside the margin, but on the correct side of the decision boundary.
  - Data points for which $\xi_n > 1$ lie on the wrong side of the decision boundary and are misclassified.

Softening the Margin



- We say the hard margin constraint is relaxed to give a soft margin that allows some of the training set data points to be misclassified.
- While slack variables allow for overlapping class distributions, the framework is still sensitive to outliers because the penalty for misclassification increases linearly with ξ.
- Our goal is now to maximize the margin while softly penalizing points that lie on the wrong side of the margin boundary.
- We minimize
$$C\sum_{n=1}^{N}\xi_n + \frac{1}{2}\|\mathbf{w}\|^2,$$
where the parameter $C > 0$ controls the trade-off between the slack variable penalty and the margin.
- Note $\sum_n \xi_n$ is an upper bound on the number of misclassified points.
- In the limit $C \to \infty$, we recover the framework for separable data.

Obtaining the Karush-Kuhn-Tucker Conditions

- The problem is to minimize $C\sum_{n=1}^{N}\xi_n + \frac{1}{2}\|\mathbf{w}\|^2$, subject to the constraints $t_n y(\mathbf{x}_n) \geq 1-\xi_n$, $n=1,\ldots,N$, and $\xi_n \geq 0$.
- The corresponding Lagrangian is

$$L(\mathbf{w},b,\mathbf{a}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N}\xi_n - \sum_{n=1}^{N}a_n\left[t_n y(\mathbf{x}_n) - 1 + \xi_n\right] - \sum_{n=1}^{N}\mu_n\xi_n,$$

  where $\{a_n \geq 0\}$ and $\{\mu_n \geq 0\}$ are Lagrange multipliers.
- The corresponding set of KKT conditions are, for $n=1,\ldots,N$,

$$\begin{array}{rcl} a_n & \geq & 0 \\ t_n y(\mathbf{x}_n) - 1 + \xi_n & \geq & 0 \\ a_n(t_n y(\mathbf{x}_n) - 1 + \xi_n) & = & 0 \\ \mu_n & \geq & 0 \\ \xi_n & \geq & 0 \\ \mu_n\xi_n & = & 0 \end{array}$$

The Dual Lagrangian

- We optimize out $\mathbf{w}$, b and $\{\xi_n\}$ making use of $y(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x}) + b$:
$$\begin{array}{lcl} \frac{\partial L}{\partial \mathbf{w}} = 0 & \Rightarrow & \mathbf{w} = \sum_{n=1}^N a_n t_n \phi(\mathbf{x}_n) \\ \frac{\partial L}{\partial b} = 0 & \Rightarrow & \sum_{n=1}^N a_n t_n = 0 \\ \frac{\partial L}{\partial \xi_n} = 0 & \Rightarrow & a_n = C - \mu_n. \end{array}$$
- Using these, we eliminate $\mathbf{w}$, b and $\{\xi_n\}$ from the Lagrangian and obtain the dual Lagrangian:
$$\widetilde{L}(\mathbf{a}) = \sum_{n=1}^N a_n - \frac{1}{2}\sum_{n=1}^N \sum_{m=1}^N a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m).$$
which is identical to the separable case, except that the constraints are somewhat different.
- Since $a_n \geq 0$ and $\mu_n \geq 0$, we get $a_n \leq C$. So minimization of $\widetilde{L}(\mathbf{a})$ with respect to $\{a_n\}$ is subject to
$$0 \leq a_n \leq C, \qquad \sum_{n=1}^N a_n t_n = 0.$$
- For predictions for new data points: $y(\mathbf{x}) = \sum_{n=1}^N a_n k(\mathbf{x}, \mathbf{x}_n) + b$.

Positioning of Support Vectors

- A subset of the data points may have $a_n = 0$, in which case they do not contribute to the predictive model $y(\mathbf{x}) = \sum_{n=1}^{N} a_n k(\mathbf{x}, \mathbf{x}_n) + b$.
- The remaining data points, with $a_n > 0$ constitute the **support vectors**.
- Support vectors satisfy $t_n y(\mathbf{x}_n) = 1 - \xi_n$.
  - If $a_n < C$, then $\mu_n > 0$, whence $\xi_n = 0$ and such points lie on the margin.
  - Points with $a_n = C$ can lie inside the margin and can
    - either be correctly classified if $\xi_n \leq 1$
    - or misclassified if $\xi_n > 1$.

Determining b

- To determine the parameter b in $y(\mathbf{x}) = \sum_{n=1}^{N} a_n k(\mathbf{x}, \mathbf{x}_n) + b$, we note that those support vectors for which $0 < a_n < C$ have $\xi_n = 0$.
- So $t_n y(\mathbf{x}_n) = 1$ and hence

$$t_n \left(\sum_{m \in \mathcal{S}} a_m t_m k(\mathbf{x}_n, \mathbf{x}_m) + b \right) = 1.$$

- A numerically stable solution is obtained by averaging to give

$$b = \frac{1}{N_{\mathcal{M}}} \sum_{n \in \mathcal{M}} \left(t_n - \sum_{m \in \mathcal{S}} a_m t_m k(\mathbf{x}_n, \mathbf{x}_m) \right),$$

  where $\mathcal{M}$ denotes the set of indices of data points having $0 < a_n < C$.
- Although predictions for new inputs are made using only the support vectors, the training phase makes use of the whole data set, and so it is important to have efficient algorithms for solving the quadratic programming problem.

The Quadratic Programming Problem

- The objective function $\widetilde{L}(\mathbf{a})$ is quadratic and so any local optimum will also be a global optimum provided the constraints define a convex region (a consequence of being linear).
- Direct solution of the quadratic programming problem using traditional techniques is often computationally infeasible.
- More practical approaches have been suggested:
  - **Chunking** (Vapnik, 1982) breaks the full problem into a series of smaller ones, exploiting the fact that the value of the Lagrangian is unchanged if we remove the rows and columns of the kernel matrix corresponding to Lagrange multipliers that have value 0.
  - **Decomposition methods** (Osuna et al., 1996) also solve a series of smaller problems but are designed so that each of these is of a fixed size, and so the technique can be applied to arbitrarily large data sets.
  - **Sequential minimal optimization**, or **SMO** (Platt, 1999) takes the concept of chunking to the extreme limit and considers just two Lagrange multipliers at a time. In this case, the subproblem can be solved analytically, avoiding numerical quadratic programming.

Subsection 4

Multiclass SVMs

Extensions to K Classes: One-versus-the-rest

- The support vector machine is a two-class classifier.
- In practice, we often have to tackle problems involving $K > 2$ classes.
- Various methods have been proposed for combining multiple two-class SVMs in order to build a multiclass classifier.
  - The **one-versus-the-rest approach** (Vapnik, 1998) constructs K separate SVMs. The k-th model $y_k(\mathbf{x})$ is trained using the data from class C_k as the positive examples and the data from the remaining $K-1$ classes as the negative examples.
    - Can lead to inconsistent results.
    - If $y(\mathbf{x}) = \max_k y_k(\mathbf{x})$ is used scaling issues arise.
    - Another problem is that the training sets are imbalanced.
  - A **variant of the one-versus-the-rest** (Lee et al., 2001) modifies the target values so that the positive class has target $+1$ and the negative class has target $\frac{-1}{K-1}$.
  - Weston and Watkins (1999) define a **single objective function** for training all K SVMs simultaneously, based on maximizing the margin from each to remaining classes. This can result in much slower training.

Extensions to K Classes: One-versus-one and DAGSVM

- Other methods used to build a multiclass classifier:
  - The **one-versus-one approach** is to train $\frac{K(K-1)}{2}$ different 2-class SVMs on all possible pairs of classes, and then to classify test points according to which class has the highest number of "votes".
    - This can lead to ambiguities in the resulting classification.
    - For large K this approach requires significantly more training time.
  - This problem can be alleviated by organizing the pairwise classifiers into a directed acyclic graph, leading to the **DAGSVM** (Platt et al., 2000). For K classes, the DAGSVM has a total of $\frac{K(K-1)}{2}$ classifiers. To classify a new test point only $K-1$ pairwise classifiers need to be evaluated depending on which path through the graph is traversed.
  - Based on error-correcting output codes, another approach, generalizing the one-versus-one approach by allowing more general partitions of the classes used to train the individual classifiers, was developed (Dietterich and Bakiri, 1995, Allwein et al., 2000). It adds robustness to errors and to ambiguity in the outputs of the individual classifiers.
- In practice the one-versus-the-rest approach is the most widely used despite its limitations.

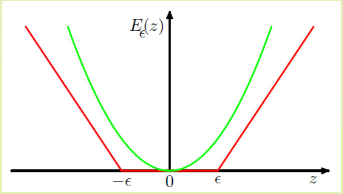
Subsection 5

SVMs for Regression

Regularized ϵ-Insensitive Error

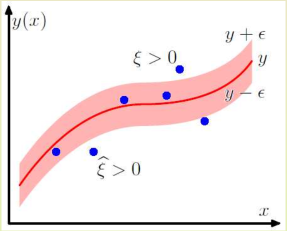
- Support vector machines can be extended to regression problems while at the same time preserving the property of sparseness.
- In simple linear regression, we minimize a regularized error function
$$\frac{1}{2}\sum_{n=1}^{N}(y_n - t_n)^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2.$$
- To obtain sparse solutions, the quadratic error function is replaced by an ϵ-**insensitive error function** (Vapnik, 1995), which gives zero error if the absolute difference between the prediction $y(\mathbf{x})$ and the target t is less than ϵ, where $\epsilon > 0$.
- A simple example of an ϵ-insensitive error function is

$$E_\epsilon(y(\mathbf{x}) - t) = \begin{cases} 0, & \text{if } |y(\mathbf{x}) - t| < \epsilon \\ |y(\mathbf{x}) - t| - \epsilon, & \text{otherwise} \end{cases}$$



The Optimization Problem

- We minimize a regularized error function
$$C\sum_{n=1}^{N} E_\epsilon(y(\mathbf{x}_n)-t_n)+\frac{1}{2}\|\mathbf{w}\|^2,$$
where $y(\mathbf{x})=\mathbf{w}^T\phi(\mathbf{x})+b$.
- By convention the (inverse) regularization parameter, denoted C, appears in front of the error term.
- For each data point $\mathbf{x}_n$, we now introduce two slack variables $\xi_n \geq 0$ and $\hat{\xi}_n \geq 0$, where



- $\xi_n > 0$ corresponds to a point for which $t_n > y(\mathbf{x}_n)+\epsilon$;
- $\hat{\xi}_n > 0$ corresponds to a point for which $t_n < y(\mathbf{x}_n)-\epsilon$.
- The condition for a target point to lie inside the ϵ-tube is that $y_n-\epsilon \leq t_n \leq y_n+\epsilon$, where $y_n = y(\mathbf{x}_n)$.

The Optimization Problem

- Introducing the slack variables allows points to lie outside the tube provided the slack variables are nonzero:
$$t_n \leq y(\mathbf{x}_n) + \epsilon + \xi_n, \quad t_n \geq y(\mathbf{x}_n) - \epsilon - \hat{\xi}_n.$$
- The error function for support vector regression is
$$C \sum_{n=1}^{N} (\xi_n + \hat{\xi}_n) + \frac{1}{2} \|\mathbf{w}\|^2$$
which must be minimized subject to $\xi_n \geq 0$, $\hat{\xi}_n \geq 0$ and the preceding two conditions.
- We introduce Lagrange multipliers $a_n \geq 0$, $\hat{a}_n \geq 0$, $\mu_n \geq 0$, $\hat{\mu}_n \geq 0$ and optimize the Lagrangian:
$$\begin{array}{rcl} L & = & C \sum_{n=1}^{N} (\xi_n + \hat{\xi}_n) + \frac{1}{2} \sum_{n=1}^{N} (\mu_n \xi_n + \hat{\mu}_n \hat{\xi}_n) \\ & & - \sum_{n=1}^{N} a_n (\epsilon + \xi_n + y_n - t_n) - \sum_{n=1}^{N} \hat{a}_n (\epsilon + \hat{\xi}_n - y_n + t_n). \end{array}$$
- Substitute $y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b$ and then set the derivatives of the Lagrangian with respect to $\mathbf{w}, b, \xi_n$ and $\hat{\xi}_n$ to zero.

The Dual Problem

- We thus get the system:

$$\begin{array}{lll} \frac{\partial L}{\partial \mathbf{w}} = 0 & \Rightarrow & \mathbf{w} = \sum_{n=1}^{N}(a_n - \hat{a}_n)\phi(\mathbf{x}_n) \\ \frac{\partial L}{\partial b} = 0 & \Rightarrow & \sum_{n=1}^{N}(a_n - \hat{a}_n) = 0 \\ \frac{\partial L}{\partial \xi_n} = 0 & \Rightarrow & a_n + \mu_n = C \\ \frac{\partial L}{\partial \hat{xi}_n} = 0 & \Rightarrow & \hat{a}_n + \hat{\mu}_n = C. \end{array}$$

- Using these results to eliminate the corresponding variables from the Lagrangian, we get the dual problem: maximize

$$\begin{array}{lll} \widetilde{L}(\mathbf{a}, \widehat{\mathbf{a}}) & = & -\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}(a_n - \hat{a}_n)(a_m - \hat{a}_m)k(\mathbf{x}_n, \mathbf{x}_m) \\ & & -\epsilon\sum_{n=1}^{N}(a_n + \hat{a}_n) + \sum_{n=1}^{N}(a_n - \hat{a}_n)t_n, \end{array}$$

  with respect to $\{a_n\}$ and $\{\hat{a}_n\}$, where $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T\phi(\mathbf{x}')$.
- To find the constraints, note that $a_n \geq 0$ and $\hat{a}_n \geq 0$, and, since $\mu_n \geq 0$ and $\hat{\mu}_n \geq 0$, we get $a_n \leq C$ and $\hat{a}_n \leq C$.
- Thus

$$0 \leq a_n \leq C, \quad 0 \leq \hat{a}_n \leq C, \quad \sum_{n=1}^{N}(a_n - \hat{a}_n) = 0.$$

Predictions and KKT Conditions

- Since $\mathbf{w} = \sum_{n=1}^{N}(a_n - \hat{a}_n)\phi(\mathbf{x}_n)$, predictions for new $\mathbf{x}$ can made using
$$y(\mathbf{x}) = \sum_{n=1}^{N}(a_n - \hat{a}_n)k(\mathbf{x}, \mathbf{x}_n) + b.$$
- The corresponding KKT conditions are
$$\begin{array}{rcl} a_n(\epsilon + \xi_n + y_n - t_n) & = & 0 \\ \hat{a}_n(\epsilon + \hat{\xi}_n - y_n + t_n) & = & 0 \\ (C - a_n)\xi_n & = & 0 \\ (C - \hat{a}_n)\hat{\xi}_n & = & 0 \end{array}$$
- Note that a_n can only be nonzero if $\epsilon + \xi_n + y_n - t_n = 0$, which implies that the data point
  - either lies on the upper boundary of the ϵ-tube ($\xi_n = 0$) or
  - lies above the upper boundary ($\xi_n > 0$).
- Similarly, a nonzero value for $\hat{a}_n$ implies $\epsilon + \hat{\xi}_n - y_n + t_n = 0$, and such points must lie
  - either on or below the lower boundary of the ϵ-tube.

Support Vectors and Evaluation of b

- The two constraints $\epsilon + \xi_n + y_n - t_n = 0$ and $\epsilon + \hat{\xi}_n - y_n + t_n = 0$ are incompatible, whence, for every $\mathbf{x}_n$, either a_n or $\hat{a}_n$ (or both) must be zero.
- The **support vectors** are those points for which $a_n \neq 0$ or $\hat{a}_n \neq 0$.
  - These are points that lie on the boundary of the ϵ-tube or outside.
  - All points within the tube have $a_n = \hat{a}_n = 0$.
- The parameter b can be found by considering a data point for which $0 < a_n < C$, i.e., since $(C - a_n)\xi_n = 0$, for which $\xi_n = 0$. Since $a_n(\epsilon + \xi_n + y_n - t_n) = 0$, they must satisfy $\epsilon + y_n - t_n = 0$.
- Solving $y(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x}) + b$ for b:

$$\begin{aligned} b &= t_n - \epsilon - \mathbf{w}^T\phi(\mathbf{x}_n) \\ &= t_n - \epsilon - \sum_{m=1}^{N}(a_m - \hat{a}_m)k(\mathbf{x}_n, \mathbf{x}_m). \end{aligned}$$

- Alternatively, we may consider a point for which $0 < \hat{a}_n < C$.
- In practice, it is better to average over all such estimates of b.