CS-667 Advanced Machine Learning

Nazar Khan

PUCIT

Support Vector Machines

Support Vector Machines

- One of the most influential machine learning techniques of the last 20 years.
- Essentially for binary classification via discriminant functions.
- Map input $\mathbf{x}$ directly to decision.
- Global optima due to convex optimization problem.
- No posterior probabilities.

Linear Classification via Discriminant Fucntions

Recap

- For 2-class linear classification with ± 1 targets, we use the linear discriminant function

$$y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}_n) + b$$

- Training: learn $\mathbf{w}^*$ and b^* from data $\mathbf{x}_1, \ldots, \mathbf{x}_N$ with targets $t_1, \ldots, t_N$.
- Testing: classify new $\mathbf{x}$ via $\text{sign}(y(\mathbf{x}))$.

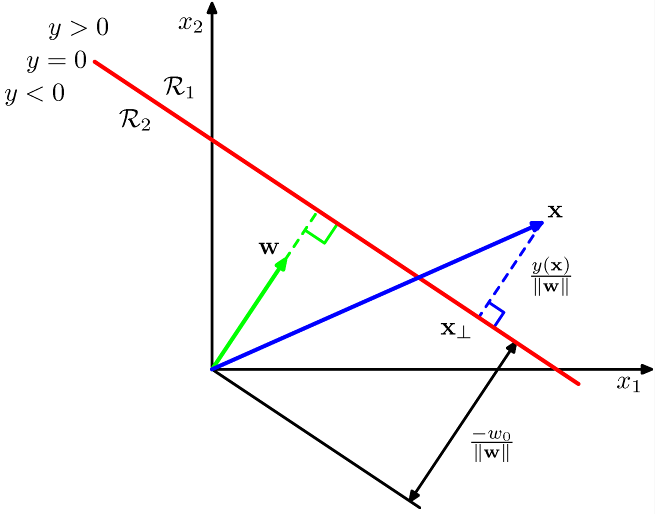
Linearly Separable Case

Maximum Margin Classifiers

- Assume dataset is linearly separable.
- That means *at least one* $\mathbf{w}, b$ configuration exists for which $y_n > 0$ for all $\mathbf{x}_n$ having $t_n = 1$ and $y_n < 0$ for all $\mathbf{x}_n$ having $t_n = -1$. That is, $t_n y_n > 0 \ \forall n$.
- Define *margin* as the distance of the closest training point from the decision surface.
- *Basic SVM idea*: choose decision surface for which *margin* is maximised.
  - If the most difficult points are maximally-separated, the rest will be separated even better.

Linearly Separable Case

Maximum Margin Classifiers



Linearly Separable Case

Maximum Margin Classifiers

- Recall from the linear classification lectures that for a decision surface $y(\mathbf{x}) = 0$
  - vector $\mathbf{w}$ is normal to the decision surface, and
  - distance of point $\mathbf{x}$ from the decision surface is given by $\frac{|y(\mathbf{x})|}{||\mathbf{w}||}$.
- For linearly separable training data $|y(\mathbf{x}_n)| = t_n y_n$ for any correct $\mathbf{w}$ and b.

Linearly Separable Case

Maximum Margin Classifiers

- So distance of training point $\mathbf{x}_n$ can be written as

$$\frac{|y(\mathbf{x}_n)|}{||\mathbf{w}||} = \frac{t_n y(\mathbf{x}_n)}{||\mathbf{w}||} = \frac{t_n \left(\mathbf{w}^T \phi(\mathbf{x}_n) + b\right)}{||\mathbf{w}||}$$

- For decision surface defined by $\mathbf{w}, b$, the margin is given by

$$\begin{aligned} \text{margin}(\mathbf{w}, b) &= \min_n \frac{t_n \left(\mathbf{w}^T \phi(\mathbf{x}_n) + b\right)}{||\mathbf{w}||} \\ &= \frac{1}{||\mathbf{w}||} \min_n t_n \left(\mathbf{w}^T \phi(\mathbf{x}_n) + b\right) \end{aligned}$$

- Optimal SVM decision boundary maximises the margin

$$\begin{aligned} \mathbf{w}^*, b^* &= \arg\max_{\mathbf{w}, b} \text{margin}(\mathbf{w}, b) \\ &= \arg\max_{\mathbf{w}, b} \left\{ \frac{1}{||\mathbf{w}||} \min_n t_n \left(\mathbf{w}^T \phi(\mathbf{x}_n) + b\right) \right\} \end{aligned}$$

Linearly Separable Case

Maximum Margin Classifiers

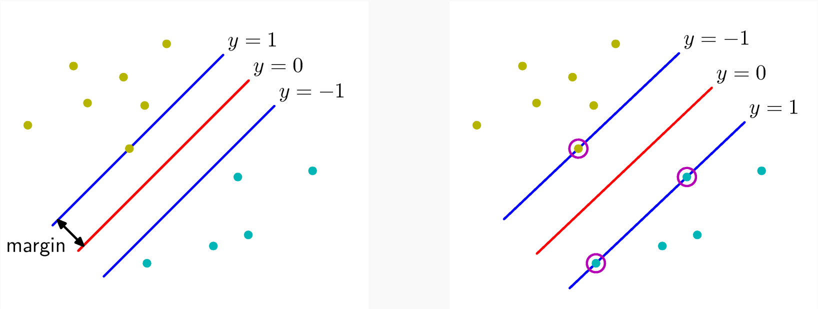


Figure: The margin is defined as the perpendicular distance between the decision boundary and the closest of the data points, as shown on the left figure. Maximizing the margin leads to a particular choice of decision boundary, as shown on the right. The location of this boundary is determined by a *subset of the data points*, known as *support vectors*, which are indicated by the circles.

Linearly Separable Case

Maximum Margin Classifiers

- Distance to boundary does not change when $\mathbf{w}$ and b are both scaled by k. (Verify this)
- Therefore, for the closest point $\mathbf{x}_c$ we can scale $\mathbf{w}$ and b by $\frac{1}{t_c\left(\mathbf{w}^T\phi(\mathbf{x}_c)+b\right)}$ *in order to set*

$$t_c\left(\mathbf{w}^T\phi(\mathbf{x}_c)+b\right)=1$$

- For all other training points $\mathbf{x}_n$, $t_n\left(\mathbf{w}^T\phi(\mathbf{x}_n)+b\right)$ will then be greater than 1.
- Therefore, we have the set of N constraints

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

- *From now on, we can redefine our margin as 1.*

Linearly Separable Case
Primal SVM Formulation

- Since $\min_n t_n \left(\mathbf{w}^T\phi(\mathbf{x}_n) + b\right) = 1$, the SVM optimisation amounts to just the maximisation

$$\mathbf{w}^*, b^* = \arg\max_{\mathbf{w},b} \frac{1}{\|\mathbf{w}\|} = \arg\min_{\mathbf{w},b} \|\mathbf{w}\|^2$$

subject to N constraints

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

which is a *quadratic programming problem*.
  - Minimisation of a *quadratic function*.
  - Subject to *linear constraints*.
- This is known as the *primal* SVM formulation.

Linearly Separable Case
Primal SVM Formulation

- Well-known solutions/packages/libraries exist for solving QP problems.
- Computational complexity of QP for M variables is $O(M^3)$.
- For high-dimensional spaces ($M > N$), a *dual* SVM formulation exists with $O(N^3)$ complexity.
- Some QP implementations solve the dual faster than the primal.
- Derivation of the dual formulation requires a thorough understanding of Lagrange multipliers.

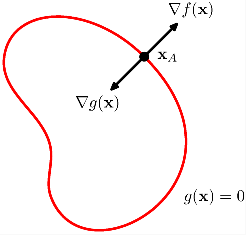
Lagrange Multipliers

- We have already seen the elegant method of *Lagrange Multipliers* for optimising functions subject to some constraints.
  1. Maximise $f(x)$ subject to *equality* constraint $g(x) = 0$.
  2. *Minimise* $f(x)$ subject to equality constraint $g(x) = 0$.
  3. Maximise $f(x)$ subject to *inequality* constraint $g(x) \geq 0$.
  4. *Minimise* $f(x)$ subject to inequality constraint $g(x) \geq 0$.
  5. Multiple constraints
- We have already covered problem 1 in CS-567.
- We will cover rest of the problems in this lecture.

Lagrange Multipliers

Problem 1: Maximisation with equality constraint

- For any surface $g(\mathbf{x}) = 0$, the gradient $\nabla g(\mathbf{x})$ is orthogonal to the surface.
- At any maximiser $\mathbf{x}^*$ of $f(\mathbf{x})$ that also satisfies $g(\mathbf{x}) = 0$, $\nabla f(\mathbf{x})$ must also be orthogonal to the surface $g(\mathbf{x}) = 0$.
  - If $\nabla f(\mathbf{x})$ is orthogonal to $g(\mathbf{x}) = 0$ at $\mathbf{x}^*$, then any movement around $\mathbf{x}^*$ along surface $g(\mathbf{x}) = 0$ is orthogonal to $\nabla f(\mathbf{x})$ and will not increase the value of f.
  - The only way to increase value of f at $\mathbf{x}^*$ is to leave the constraint surface $g(\mathbf{x}) = 0$.



- So, at any maximiser $\mathbf{x}^*$, ∇f and ∇g are parallel (or anti-parallel) vectors.
- This can be stated mathematically as

$$\nabla f + \lambda \nabla g = 0$$

where $\lambda \neq 0$ is the so-called *Lagrange multiplier*.
- This can also be formulated as the *unconstrained* maximisation of the so-called *Lagrangian function*

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

with respect to $\mathbf{x}$ and λ.

Lagrange Multipliers

Problem 2: Minimisation with equality constraint

- Minimisation of $f(\mathbf{x})$ is equivalent to maximisation of $-f(\mathbf{x})$.
- At any maximiser $\mathbf{x}^*$ of $-f(\mathbf{x})$, we will have

$$-\nabla f + \lambda \nabla g = 0$$

- This corresponds to unconstrained maximisation of

$$-f(\mathbf{x}) + \lambda g(\mathbf{x})$$

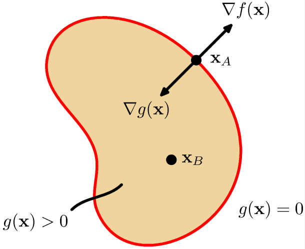
  or equivalently the unconstrained minimisation w.r.t $\mathbf{x}$ of the Lagrangian

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

Lagrange Multipliers

Problem 3: Maximisation with inequality constraint

- When the constraint $g(\mathbf{x}) \geq 0$, $\mathbf{x}^*$ can be either
  1. *on* the constraint surface (active constraint $g(\mathbf{x}) = 0$), or
  2. *within* the constraint surface (inactive constraint $g(\mathbf{x}) > 0$)
- Case 1 with $g(\mathbf{x}) = 0$ implies $\lambda > 0$ since ∇f must be anti-parallel. (Why anti-parallel?)
- Case 2 with $g(\mathbf{x}) > 0$ does not constrain the direction of ∇f. All that is required from a maximiser $\mathbf{x}^*$ is $\nabla f|_{\mathbf{x}^*} = 0$ which implies $\lambda = 0$.



- Combining both cases, we have three conditions

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

- These three conditions are known as the *Karush-Kuhn-Tucker* (KKT) conditions for optimisation with inequality constraints.
- So the unconstrained maximisation uses the Lagrangian function

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

and satisfies the three KKT conditions.

Lagrange Multipliers

Problem 4: Minimisation with inequality constraint

- Corresponds to unconstrained *minimisation w.r.t* $\mathbf{x}$ and *maximisation w.r.t* λ of the Lagrangian function

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

and satisfies the three KKT conditions.

Lagrange Multipliers
Problem 5: Multiple constraints

- For maximisation with K constraints, the Lagrangian uses K Lagrange multipliers λ_k and is written as

$$L(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \sum_{k=1}^{K} \lambda_k g_k(\mathbf{x})$$

Dual SVM Formulation

- The SVM problem *minimises* $\frac{1}{2}\|\mathbf{w}\|^2$ subject to N *inequality* constraints of the form $t_n\left(\mathbf{w}^T\phi(\mathbf{x}_n)+b\right)-1\geq 0$.
- The Lagrangian function can be written as

$$L(\mathbf{w},b,\mathbf{a})=\frac{1}{2}\|\mathbf{w}\|^2-\sum_{n=1}^{N}a_n\left\{t_n\left(\mathbf{w}^T\phi(\mathbf{x}_n)+b\right)-1\right\}$$

  where $a_n\geq 0$ are the N Lagrange multipliers.
- The KKT conditions can be written as

$$\begin{aligned} a_n &\geq 0 \\ t_n\left(\mathbf{w}^T\phi(\mathbf{x}_n)+b\right)-1 &\geq 0 \\ a_n\left\{t_n\left(\mathbf{w}^T\phi(\mathbf{x}_n)+b\right)-1\right\} &= 0 \end{aligned}$$

- Setting the gradients of the Lagrangian to zero

$$\mathbf{0} \equiv \frac{\partial L}{\partial \mathbf{w}} \implies \mathbf{w} = \sum_{n=1}^{N} a_n t_n \phi(\mathbf{x}_n)$$

$$0 \equiv \frac{\partial L}{\partial b} \implies \sum_{n=1}^{N} a_n t_n = 0$$

- By replacing these two conditions in the Lagrangian, we can *eliminate* $\mathbf{w}$ and b to obtain the *dual* SVM formulation in just the N variables a_n.
- Take-home Quiz 4: Show that by eliminating $\mathbf{w}$ and b from the Lagrangian $L(\mathbf{w}, b, \mathbf{a})$, we obtain the expression for the dual

$$\tilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m \phi(\mathbf{x}_n)^T \phi(\mathbf{x}_m)$$

- The *dual* formulation of the max-margin SVM problem is the maximisation of

$$\tilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m \underbrace{\phi(\mathbf{x}_n)^T\phi(\mathbf{x}_m)}_{k(\mathbf{x}_n,\mathbf{x}_m)}$$

w.r.t $\mathbf{a}$ subject to the $N+1$ constraints

$$a_n \geq 0, \quad n = 1, \ldots, N$$

$$\sum_{n=1}^{N} a_n t_n = 0$$

- This is once again a QP problem but in N variables with complexity $O(N^3)$.

The Kernel Trick

- Scalar product $\phi(\mathbf{x}_n)^T\phi(\mathbf{x}_m)$ measures similarity in feature space $\phi(\cdot)$.
- Similarity can be also be measured by alternative functions. For example, Euclidean distance between $\mathbf{x}_n$ and $\mathbf{x}_m$.
- *The Kernel Trick*: Replace scalar product by some other, more suitable *kernel* function $k(\mathbf{x}_n, \mathbf{x}_m)$.
  - Also known as *kernel substitution*.
  - **This is what gives SVMs the flexibility to be applied to many different kinds of problems**.
  - For example, we can have kernels like k(web page 1, web page 2), k(document 1, document 2), k(DNA sequence 1,DNA sequence 2), k(sentence 1, sentence 2), $\cdots$.

- If we have the kernel value $k(\mathbf{x}_n, \mathbf{x}_m)$, we don't even need to compute feature $\phi(\mathbf{x})$.
  - Allows us to work in very high (even infinite) dimensional feature spaces.
- *Any* algorithm (not just SVMs) in which inputs appear only in terms of scalar products, can be made more powerful by replacing the scalar products with more powerful, problem-specific kernel functions.
  - Kernel linear regression.
  - Kernel PCA.

Dual SVM Formulation

- Notice that by moving to the dual formulation, we have sacrificed the parametric nature of the primal formulation.
- This means that in the dual formulation, we need all the training data at test time.
- This is similar to nearest-neighbour classifiers, Parzen windows based density estimation, *etc*.
- However, SVMs require only a subset of the training data – the so-called *support vectors*.
- So we get the best of both worlds!

Support Vectors

- The classifier output can be written as

$$
\begin{aligned}
y(\mathbf{x}) &= \mathbf{w}^T\phi(\mathbf{x}) + b \\
&= \sum_{n=1}^{N} a_n t_n \underbrace{\phi(\mathbf{x}_n)^T\phi(\mathbf{x})}_{k(\mathbf{x}_n,\mathbf{x})} + b
\end{aligned}
$$

- All data points $\mathbf{x}_n$ for which $a_n = 0$ have no role in determining the classifier's output.
- Therefore, we *only need to store the training data points for which $a_n > 0$*.
- These data points are called the *support vectors*.

$$
y(\mathbf{x}) = \sum_{m\in\mathcal{S}} a_m t_m k(\mathbf{x}_n, \mathbf{x}_m) + b
$$

where $\mathcal{S}$ is the set of indices of the support vectors.

Determining b

- From the KKT conditions, we know that for any support vector, *i.e.* $a_n > 0$, we must have

$$t_n \left(\mathbf{w}^T \phi(\mathbf{x}_n) + b \right) = 1$$

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

- Multiplying both sides by t_n and using the fact that $t_n^2 = 1$, we obtain an estimate for b

$$b = t_n - \sum_{m \in \mathcal{S}} a_m t_m k(\mathbf{x}_n, \mathbf{x}_m)$$

- A better estimate for b can be obtained by averaging over all support vectors

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

Kernels

- Linear kernels $k(\mathbf{x}, \mathbf{x}_0) = \mathbf{x}^T \mathbf{x}_0$.
- Polynomial kernels $k(\mathbf{x}, \mathbf{x}_0) = (1 + \mathbf{x}^T \mathbf{x}_0)^d$ for any $d > 0$.
  - Contains all polynomial terms up to degree d.
- Gaussian kernels $k(\mathbf{x}, \mathbf{x}_0) = \exp\left(\frac{-||\mathbf{x}-\mathbf{x}_0||^2}{2\sigma^2}\right)$ for $\sigma > 0$.
  - Infinite dimensional feature space.
- `https://youtu.be/XUj5JbQihlU?t=812`

Summary

- Data may be linearly separable in a high dimensional feature space ϕ, but not in the input space $\mathbf{x}$.
- Classifiers can be learnt for this high dimensional feature space without actually computing $\phi(\mathbf{x})$.
- Kernel trick replaces the scalar product in the dual formulation.
- Kernel trick can be used in other ML approaches.
- Kernels can be applied to a large variety of objects (not just vectors).
- So far: linearly separable data. Next we discuss SVMs for non-separable data.

Linearly Non-Separable Case

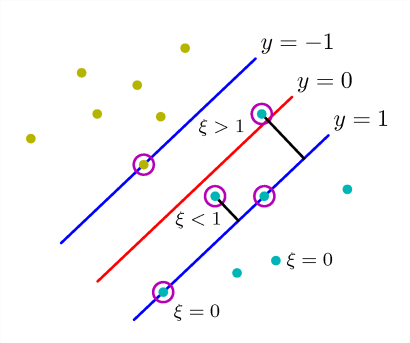
- Assume data is linearly non-separable.
- We can still learn a linear decision boundary in ϕ-space corresponding to a non-linear one in $\mathbf{x}$-space.
- However, such exact non-linear separation of training data can lead to over-fitting.
- It can be a good idea to *allow some misclassifications* of the training points.

Slack Variables

- This is achieved by replacing the *hard margin constraints* $t_n y_n \geq 1$ by *soft margin constraints* $t_n y_n + \xi_n \geq 1$ where $\xi_n \geq 0$.
- The addition of the *slack variables* ξ_n allows $t_n y_n$ to be less than 1 and still satisfy the soft margin constraint.
- If hard constraint $t_n y_n \geq 1$ is not being satisfied, we *help* by adding ξ_n in order to reach 1.
- ξ_n represents the minimum amount to be added to make $t_n y_n + \xi_n = 1$.

Slack Variables

$\xi_n = 0$	correctly classified either on or on the correct side of the margin
$0 < \xi_n < 1$	correctly classified within the margin
$\xi_n = 1$	on the decision surface
$\xi_n > 1$	misclassified



SVM with Soft Margin Costraints

- Goal: Maximise margin while softly penalising points that lie on the wrong side of the margin.
- Achieved via

$$\arg\min_{\mathbf{w},b,\xi_1,\dots,\xi_N} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N}\xi_n$$

$$\text{s.t. } t_n y_n + \xi_n \geq 1 \text{ for } n = 1,\dots,N$$

$$\xi_n \geq 0 \text{ for } n = 1,\dots,N$$

- Parameter $C > 0$ controls the trade-off between misclassifications and maximising the margin.
  - Large $C \implies$ penalising slack $\implies$ good training performance $\implies$ over-fitting.
  - Small C allows misclassifications on training data.
  - So C is like an inverse-regularisation parameter.
- The sum $\sum_{n=1}^{N}\xi_n$ is an upper-bound on the number of misclassifications. (Why?)

Dual Formulation

- We have a constrained minimisation problem with inequality constraints.
- Lagrangian can be written as

$$L(\mathbf{w}, b, \mathbf{a}, \boldsymbol{\mu}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N}\xi_n - \underbrace{\sum_{n=1}^{N} a_n\left\{t_n y_n + \xi_n - 1\right\}} - \underbrace{\sum_{n=1}^{N}\mu_n\xi_n}$$

$$\begin{aligned} a_n &\geq 0 & \mu_n &\geq 0 \\ t_n y_n + \xi_n - 1 &\geq 0 & \xi_n &\geq 0 \\ a_n\left\{t_n y_n + \xi_n - 1\right\} &= 0 & \mu_n\xi_n &= 0 \end{aligned}$$

where $a_n \geq 0$ are Lagrange multipliers for the N soft margin constraints and $\mu_n \geq 0$ are Lagrange multipliers for the N slack variable constraints.

Dual Formulation

- The $6N$ KKT conditions can be written as

$$a_n \geq 0$$
$$t_n y_n + \xi_n - 1 \geq 0$$
$$a_n \left\{ t_n y_n + \xi_n - 1 \right\} = 0$$
$$\mu_n \geq 0$$
$$\xi_n \geq 0$$
$$\mu_n \xi_n = 0$$

Dual Formulation

- Similar to the separable case, we can set

$$\mathbf{0} \equiv \frac{\partial L}{\partial \mathbf{w}} \implies \mathbf{w} = \sum_{n=1}^{N} a_n t_n \phi(\mathbf{x}_n)$$

$$0 \equiv \frac{\partial L}{\partial b} \implies \sum_{n=1}^{N} a_n t_n = 0$$

$$0 \equiv \frac{\partial L}{\partial \xi_n} \implies a_n = C - \mu_n$$

to optimise out (eliminate)

  - the original parameters $\mathbf{w}, b$,
  - the slack variables ξ_n, and
  - Lagrange multipliers μ_n

Dual Formulation

- This yields the dual formulation

$$\tilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m \underbrace{\phi(\mathbf{x}_n)^T \phi(\mathbf{x}_m)}_{k(\mathbf{x}_n, \mathbf{x}_m)}$$

- The constraints that carry over are $a_n \geq 0$ and $\sum_{n=1}^{N} a_n t_n = 0$.
- Since $a_n = C - \mu_n$ and $\mu_n \geq 0$, we must have $a_n \leq C$.
- So the $N + 1$ constraints become

$$0 \leq a_n \leq C, \quad n = 1, \ldots, N \qquad (\textit{box constraints})$$

$$\sum_{n=1}^{N} a_n t_n = 0$$

- Once again, we have a QP problem in N variables.

Dual Formulation

- After solving the QP problem for $\mathbf{a}^*$, we get $a_n = 0$ for some data points. These points play no role during predictions for arbitrary $\mathbf{x}$.
- For the remaining points (*i.e.*, support vectors), we have 2 cases:
  1. $a_n < C \implies \mu_n > 0 \implies \xi_n = 0 \implies \mathbf{x}_n$ lies on (or beyond) margin.
  2. $a_n = C \implies \mu_n = 0 \implies \xi_n > 0$ which in turn yields 3 cases
     - 2.1 $\xi_n < 1 \implies \mathbf{x}_n$ lies within the margin but correctly classified.
     - 2.2 $\xi_n = 1 \implies \mathbf{x}_n$ lies on the decision surface.
     - 2.3 $\xi_n > 1 \implies \mathbf{x}_n$ is misclassified.
- A popular technique for SVM training is *sequential minimal optimisation (SMO)* which avoids quadratic programming.
- Scales between $O(N)$ and $O(N^2)$.

Multiclass SVMs

- An SVM is fundamentally a binary classifier.
- Can be trained for multiclass problems via
  - One-versus-rest approach. Leads to ambiguous classification regions, imbalanced datasets, differing output scales.
  - One-vs-one approach. Leads to ambiguous classification regions and slower training and testing.
- One-vs-rest approach is used more often.

Extensions
Structured Outputs

- Structured output variables have dependencies between each other.
  - Images, trees, DNA sequences, *etc*.
- *Structural SVMs* have been developed for such structured output spaces.
- Similar max-margin framework can be used.
- Tsochantaridis I, Hofmann T, Joachims T, Altun Y (2004) Support vector machine learning for interdependent and structured output spaces. In: International Conference on Machine Learning (ICML), pp 104–112

Extensions
Others

- Regression problems can be addressed by *Support Vector Regression (SVR)*.
- Posterior probabilities are output by a *Relevance Vector Machine (RVM)*.

Mid-term Exam

- Take-home quizzes.
- Blue points in lecture slides.
- Everything else in lecture slides.
- Practical things you learned while completing the projects.