Machine Learning
Week 9 Lecture:- Support Vector Machines, Kernelisation

Debasis Ganguly
Debasis.Ganguly@glasgow.ac.uk

School of Computing Science
University of Glasgow

November 28, 2025

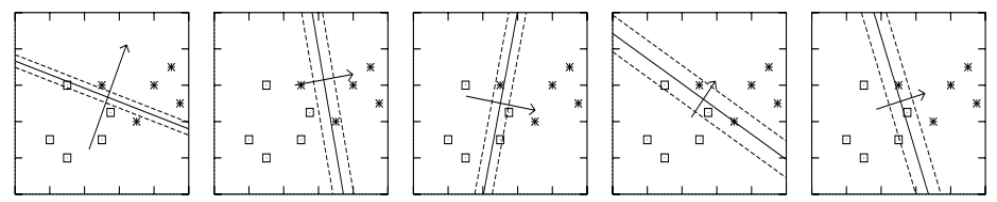
Introduction
D. Ganguly
Support Vector Machine
Margin
Maximising the margin
The Support Vector Machine
Soft margins
Kernels
Summary

Pros and Cons of Bayesian

Bayesian approaches

Allows modeling uncertainties in predictions via

- Not relying too much on one boundary.
- Aggregating beliefs over several boundaries.



Pros

- A post-processing method.
- **Adjusts** the predictions obtained with a point estimate (gradient descent).
- So, this means **no change required in the objective function**.

Cons

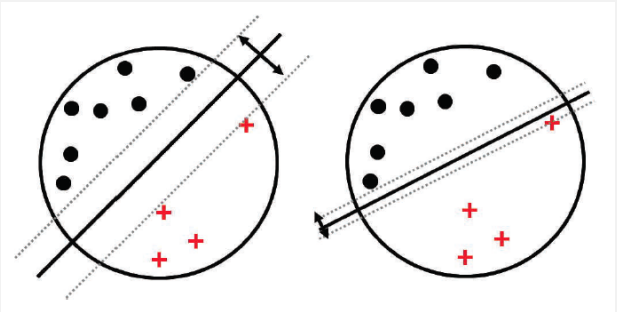
- No direct modeling of the uncertainty into the objective function itself.
- Is likely to not work well for high dimensional data.

Maximizing margin width

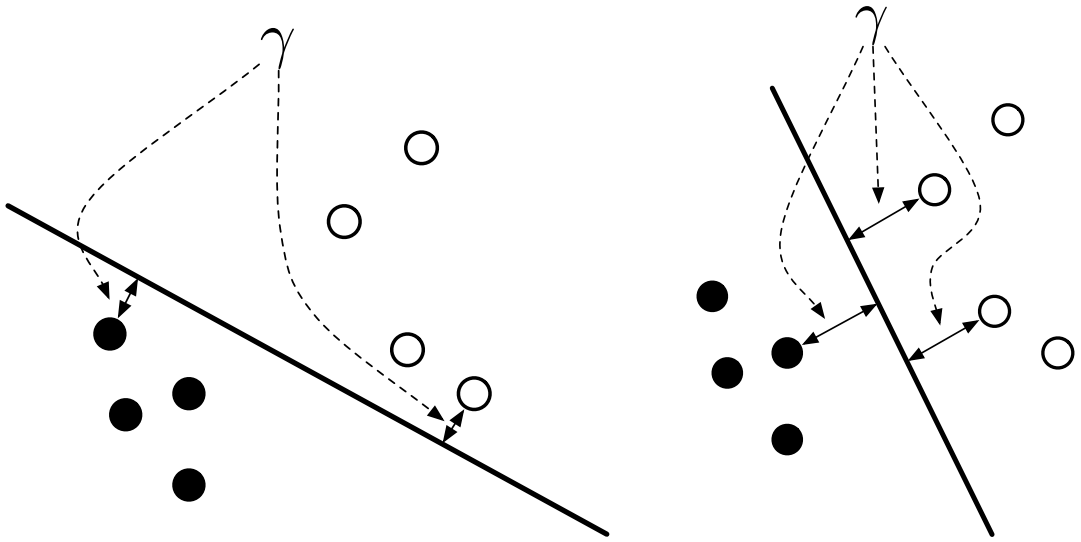
- Logistic Regression: Minimizes cross-entropy loss.
- Naive Bayes: Maximizes posterior class likelihood given the prior class conditionals.
- Bayesian approaches: Maximizes posterior likelihood with a local Gaussian fit (variational), or maximizes posterior likelihood of samples.

Support Vector Machines

- Finds the decision boundary that **maximises the margin**.
- Intuition: Keeps **as much provision as possible for errors** (in generalisation). So, a different take on tackling uncertainty as compared to Bayesian.



Why maximise the margin?

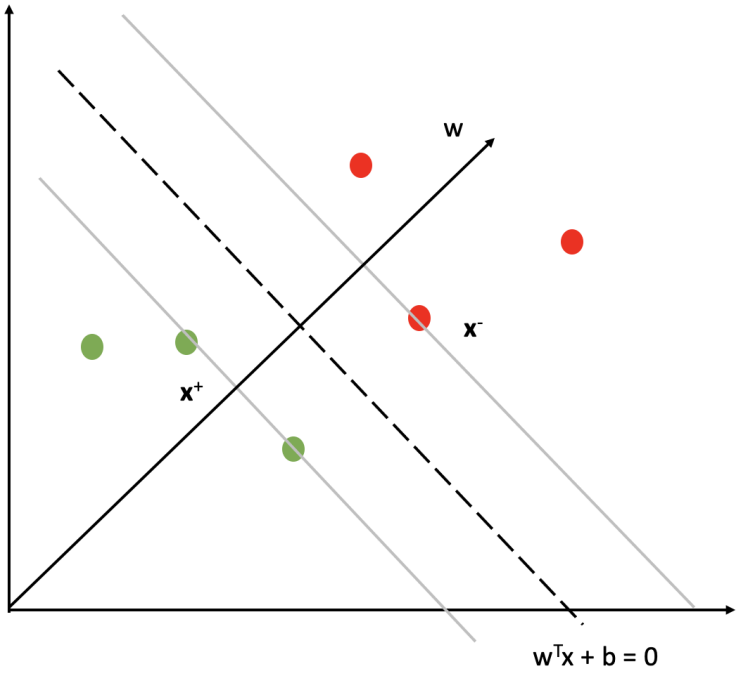


For SVMs

- Decision boundary: $\mathbf{w}^T\mathbf{x} + b = 0$
- Constraints: $|\mathbf{w}^T\mathbf{x} + b| \geq 1 \ \forall \mathbf{x}$

- **Left**: A classifier (e.g., logistic regression) that doesn't explicitly take into account margin width.
- **Right**: Maximum margin decision boundary seems to better reflect the data characteristics.
- How to obtain which boundary leads to the maximum margin?

Computing the margin width

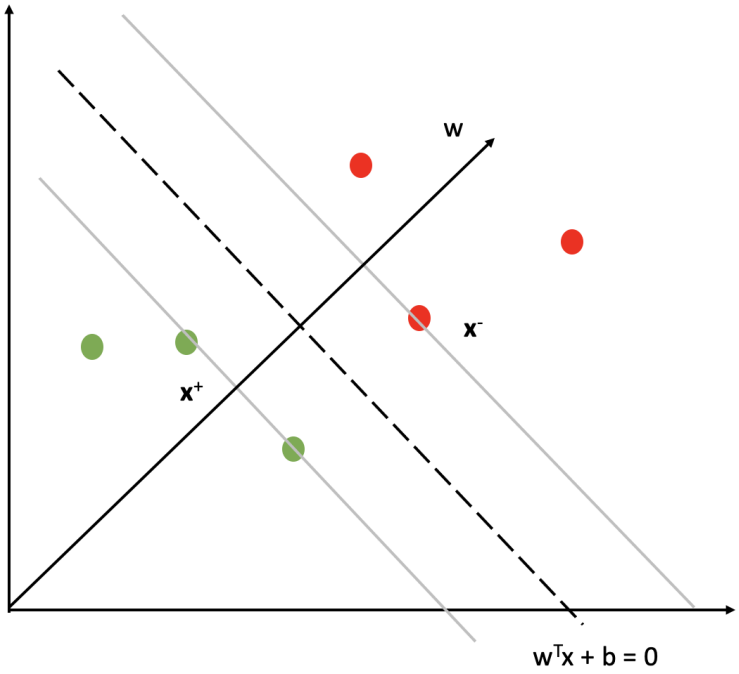


- **w** - A vector perpendicular to the decision boundary $\mathbf{w}^T\mathbf{x} + b$.
- Unit vector: $\mathbf{w}/||\mathbf{w}||$.

Menti-Quiz (Code: 2875-9119)

Why is the parameter vector $\mathbf{w}^T$ orthogonal to the decision boundary $\mathbf{w}^T\mathbf{x} + b$?

Computing the margin width



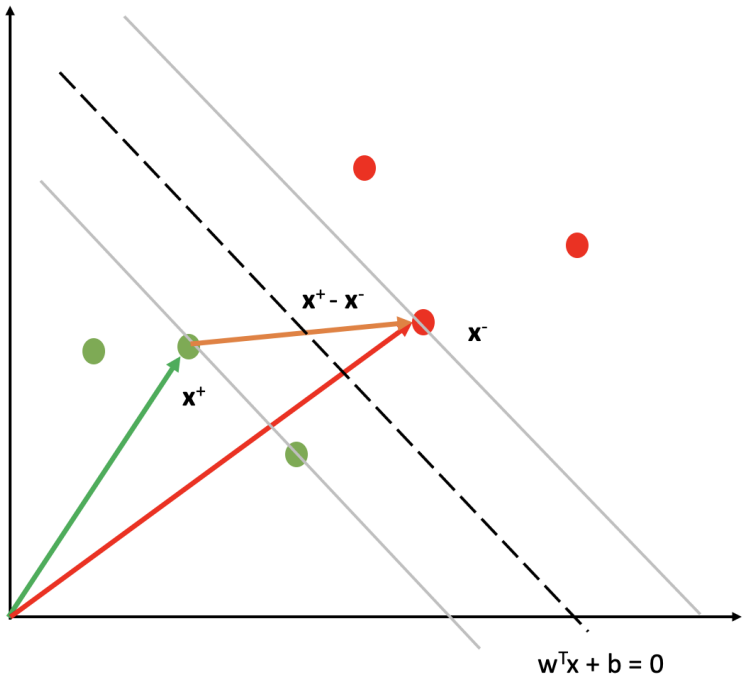
- **w** - A vector perpendicular to the decision boundary $\mathbf{w}^T\mathbf{x} + b$.
- Unit vector: $\mathbf{w}/||\mathbf{w}||$.

Menti-Quiz (Code: 2875-9119)

Why is the parameter vector $\mathbf{w}^T$ orthogonal to the decision boundary $\mathbf{w}^T\mathbf{x} + b$?

- By definition, two vectors **a** and **b** are orthogonal if $\mathbf{a} \cdot \mathbf{b} = 0 = \sum_i a_i b_i$.
- We know that for any point **x** on the boundary, $\mathbf{w}^T\mathbf{x} = -b$
- This implies **w** is orthogonal to $\mathbf{w}^T\mathbf{x} + b$ and also orthogonal to any line parallel to it.

Computing the margin width

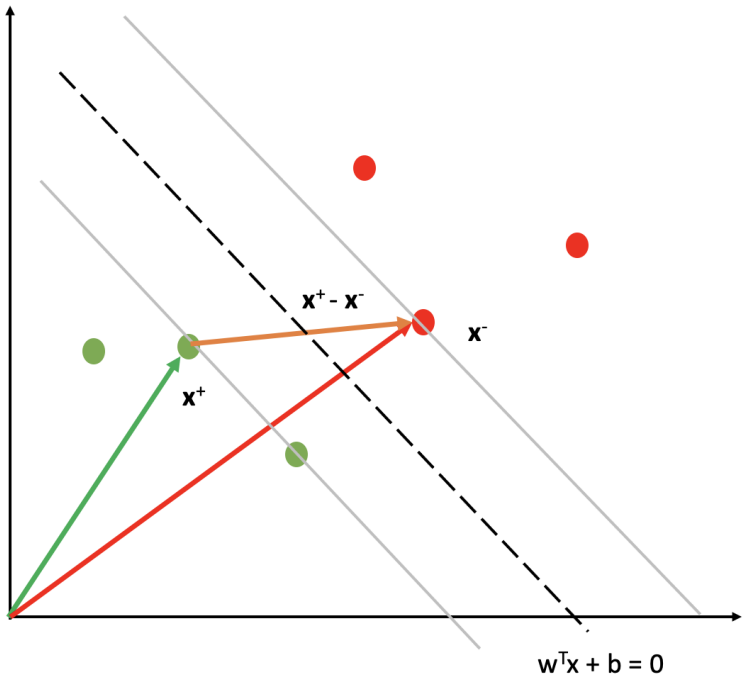


- Consider the difference vector $\mathbf{x}_+$ - $\mathbf{x}_-$.
- Projection of the difference vector ($\mathbf{x}_+$ - $\mathbf{x}_-$) on the unit vector orthogonal to the decision boundary?

Menti-Quiz (Code: 2875-9119)

Try to draw the projection vector yourself.

Computing the margin width

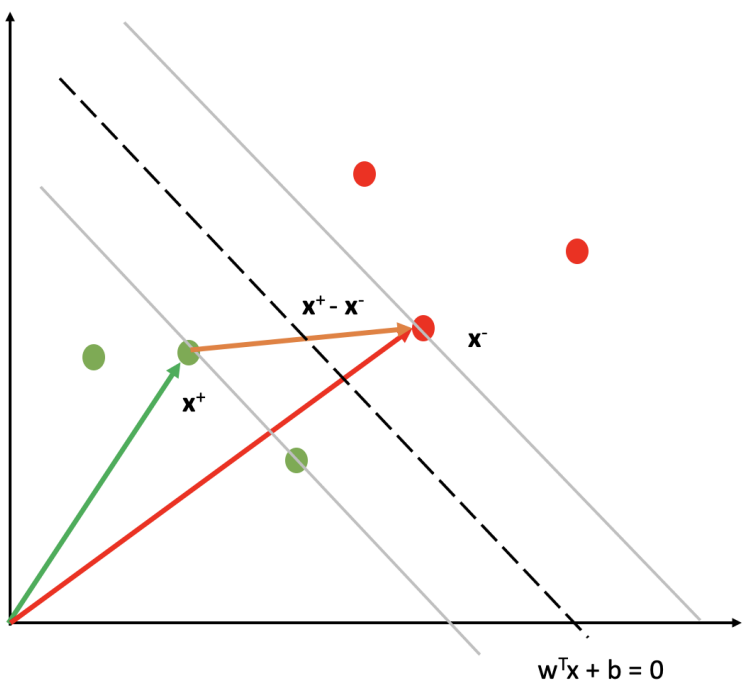


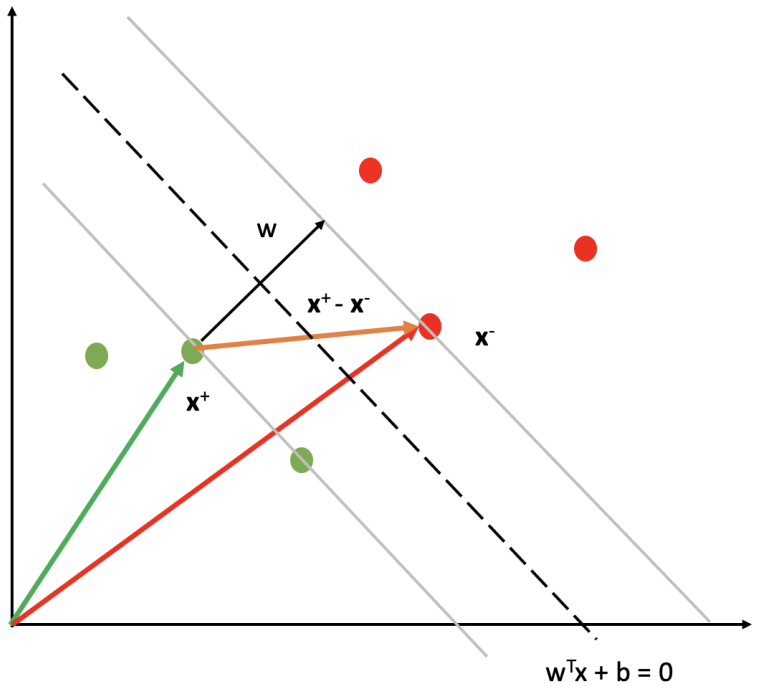
- Consider the difference vector $\mathbf{x}_+$ - $\mathbf{x}_-$.
- Projection of the difference vector ($\mathbf{x}_+$ - $\mathbf{x}_-$) on the unit vector orthogonal to the decision boundary?

Menti-Quiz (Code: 2875-9119)

Try to draw the projection vector yourself.

Computing the margin width



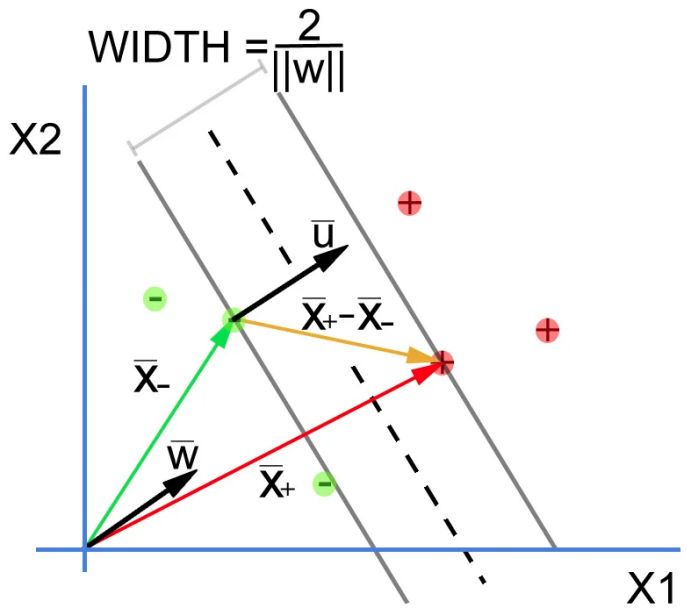


- Margin width: Projection of the difference vector ($\mathbf{x}_+$ - $\mathbf{x}_-$) on the unit vector $\frac{1}{||\mathbf{w}||}\mathbf{w}$.
- Projection of a vector **a** onto **b** is: the vector **b scaled by the dot product**: $(\frac{\mathbf{a}.\mathbf{b}}{||\mathbf{b}||})\mathbf{b}$
- $\gamma = \frac{1}{||\mathbf{w}||}\mathbf{w}^T(\mathbf{x}_+ - \mathbf{x}_-)$

Computing the margin width

- $\gamma = \frac{1}{||\mathbf{w}||} \mathbf{w}^T(\mathbf{x}_+ - \mathbf{x}_-)$

$$\begin{aligned} \mathbf{w}^T\mathbf{x}_+ + b &= 1 \\ \mathbf{w}^T\mathbf{x}_- + b &= -1 \end{aligned}$$

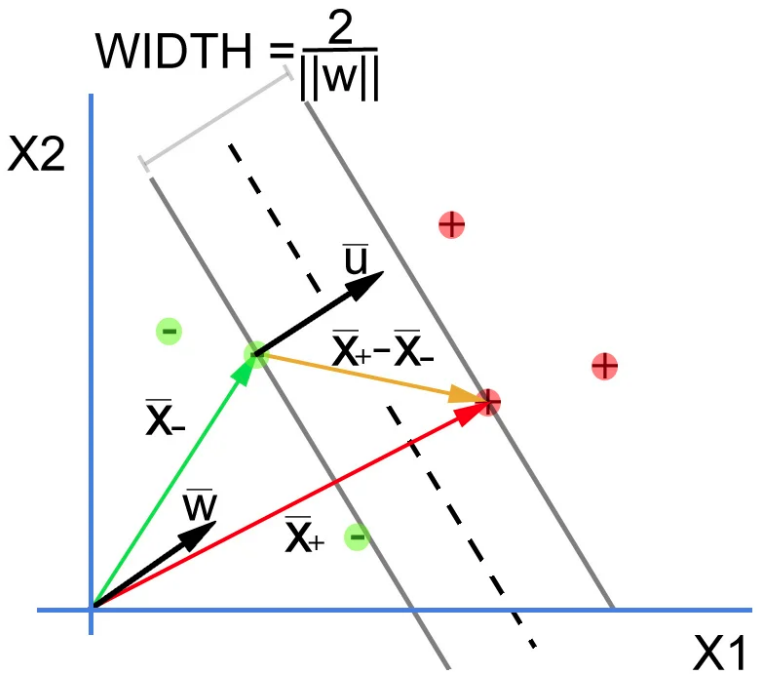


- This means:
- $(\mathbf{w}^T\mathbf{x}_+ + b) - (\mathbf{w}^T\mathbf{x}_- + b) = 2$
- Substituting:
- $\mathbf{w}^T\mathbf{x}_+ - \mathbf{w}^T\mathbf{x}_- = 2$
- $\gamma = \frac{2}{||\mathbf{w}||}$

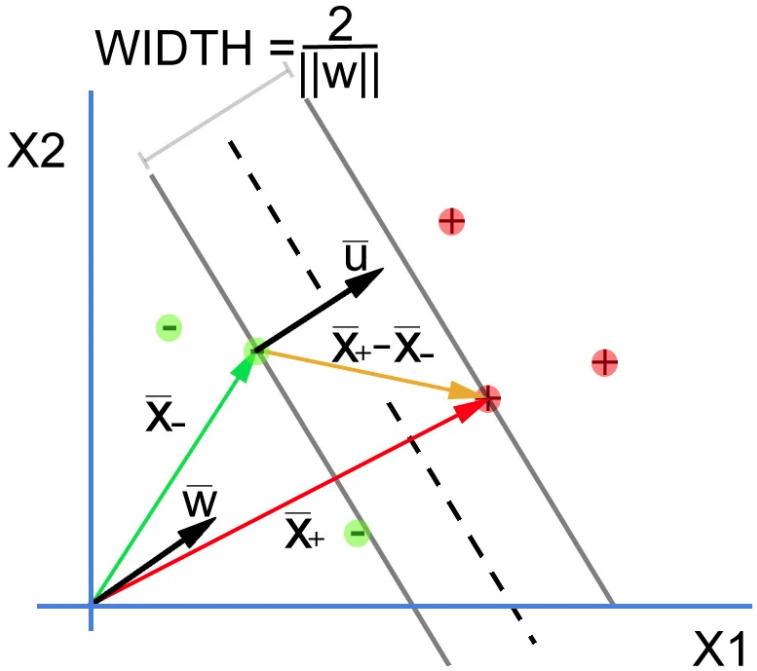
Maximising the margin

- We want to maximise $\gamma = \frac{2}{||\mathbf{w}||}$
- Equivalent to minimising $||\mathbf{w}||$
- Equivalent to minimising $\frac{1}{2}||\mathbf{w}||^2 = \frac{1}{2}\mathbf{w}^T\mathbf{w}$

- There are some constraints (what are they?):



Maximising the margin



- For $\mathbf{x}_n$ with $t_n = 1$: $\mathbf{w}^T\mathbf{x}_n + b \geq 1$
- For $\mathbf{x}_n$ with $t_n = -1$: $\mathbf{w}^T\mathbf{x}_n + b \leq -1$
- Can be expressed more neatly as:

Constraints in an SVM objective

$$t_n(\mathbf{w}^T\mathbf{x}_n + b) \geq 1$$

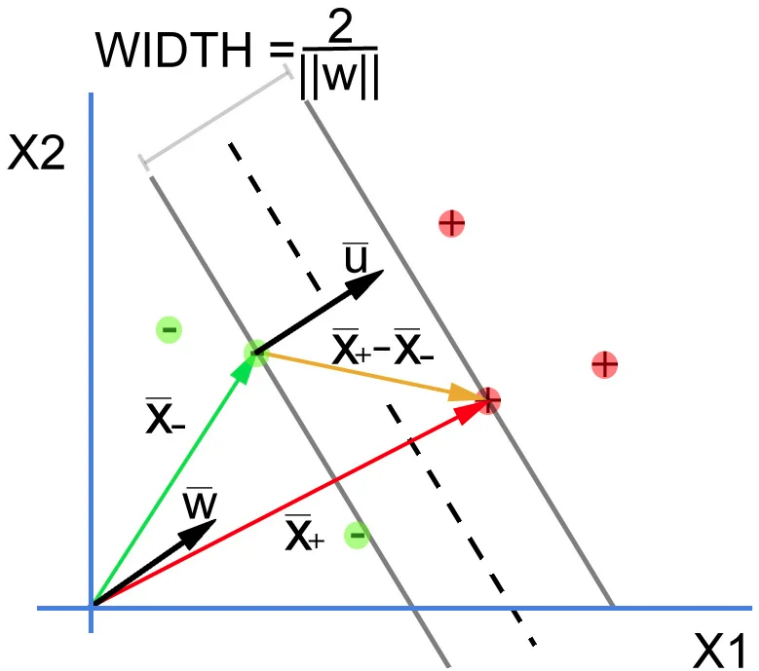
- (This is why we use $t_n = \pm 1$ and not $t_n = \{0, 1\}$ as in Logistic Regression)

Maximising the margin

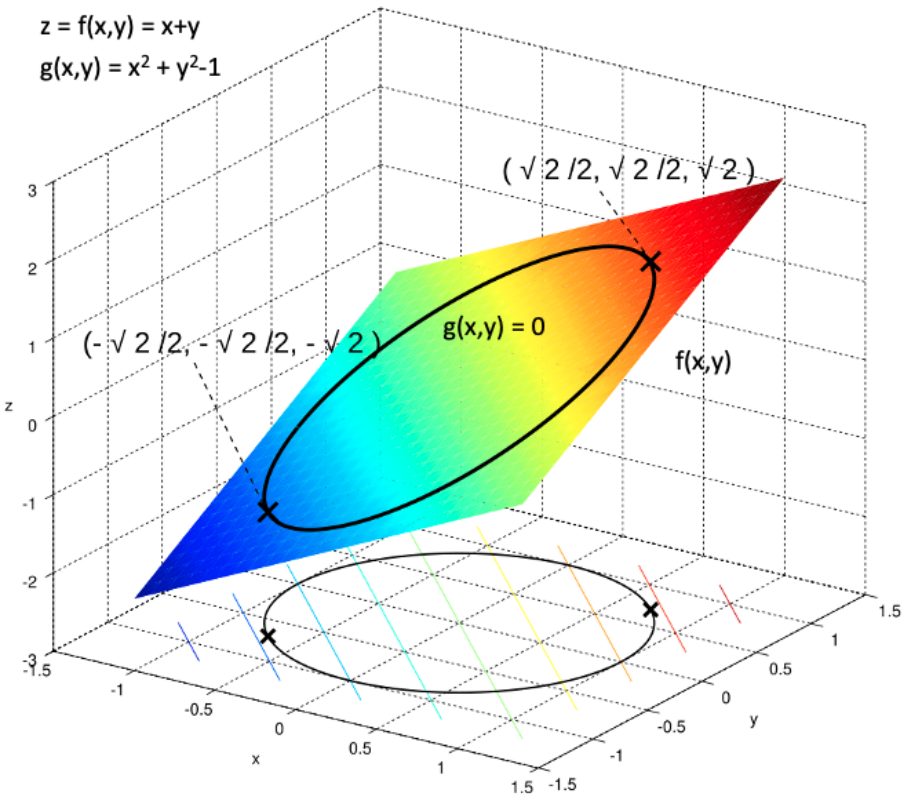
SVM Objective (Primal)

$$\underset{\mathbf{w}}{\text{argmin}} \ \frac{1}{2}\mathbf{w}^T\mathbf{w}$$

$$\text{Subject to: } t_n(\mathbf{w}^T\mathbf{x}_n + b) \geq 1$$



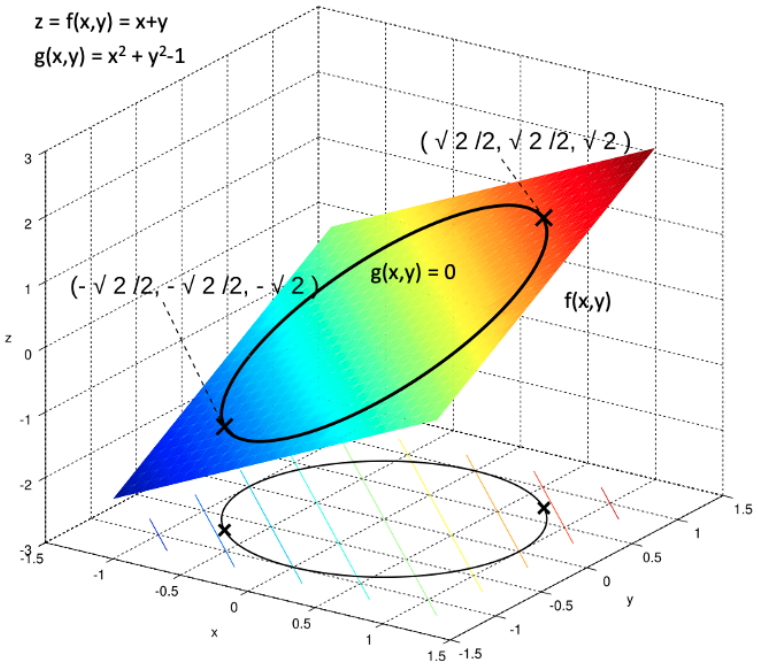
Lagrangian Duals



- The solution is a stationary point w.r.t. parameters and the Lagrangian coefficient, i.e., at the solution point $\nabla \mathcal{L} = 0$.
- Otherwise, one might walk along $g(x, y) = c$ and reach a higher value (contradictory).

- Maximization objective $f(x, y)$ s.t. $g(x, y) = c$
- **Changed** to: Maximize $\mathcal{L}(x, y, \alpha) = f(x, y) + \alpha(g(x, y) - c)$, s.t., $\alpha > 0$
  - α is a new **learnable** parameter (not a hyper-parameter)

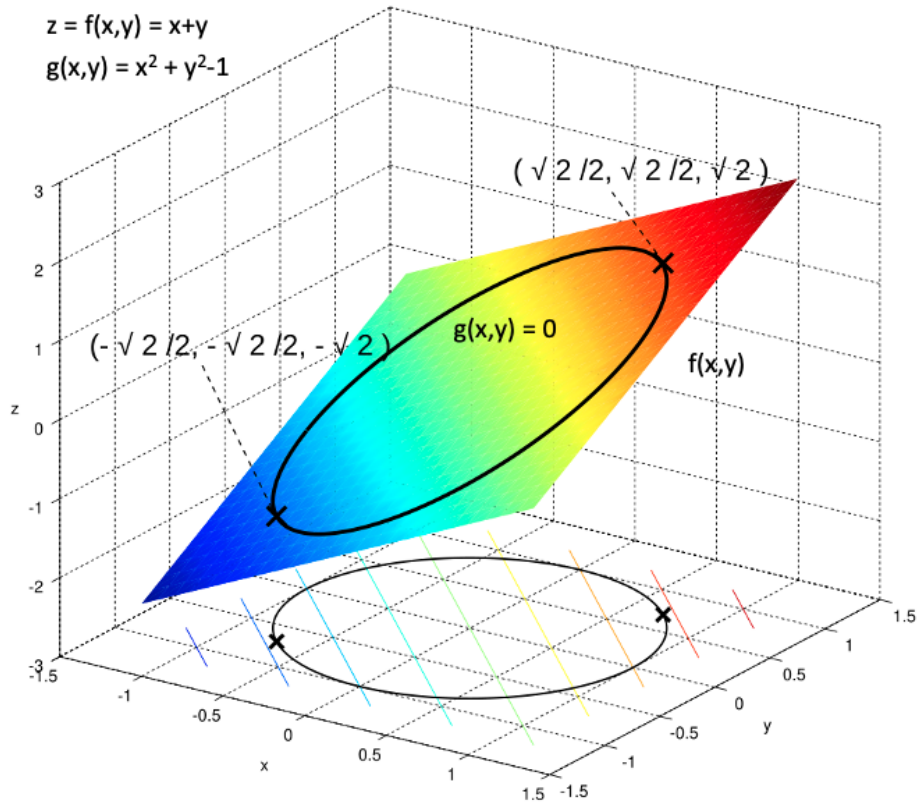
Lagrangian Duals



- $\mathcal{L}(x, y, \alpha) = f(x, y) + \alpha(g(x, y) - c)$
- Constraint is **not just penalised**.
- Rather it is **forced**.

- Intuitively, feels similar to loss with penalisation: $\mathcal{L}(\theta, \mathbf{x}) = (y - h_\theta(\mathbf{x}))^2 + ||\theta||$
- Intuitively, $\alpha(g(x, y) - c)$ adds a penalisation to $f(x, y)$. But this is **not the entire story**.

Lagrangian Duals



Menti-Quiz (Code: 2875-9119)

Why is the constraint enforced?

Introduction

D. Ganguly

Support Vector Machine

Margin

Maximising the margin

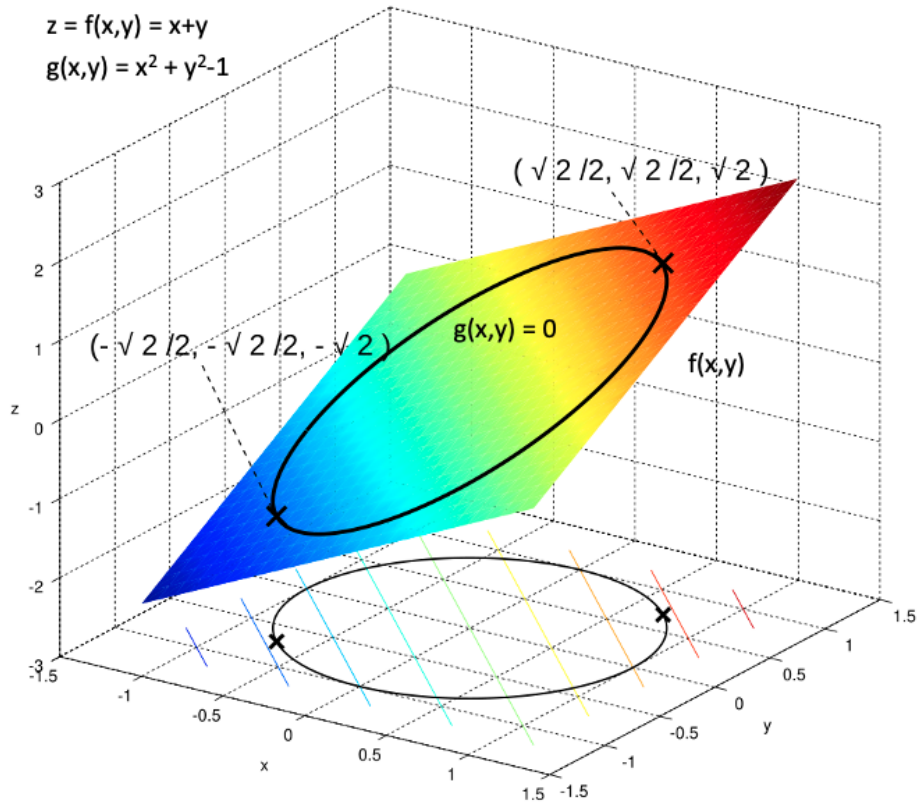
The Support Vector Machine

Soft margins

Kernels

Summary

Lagrangian Duals

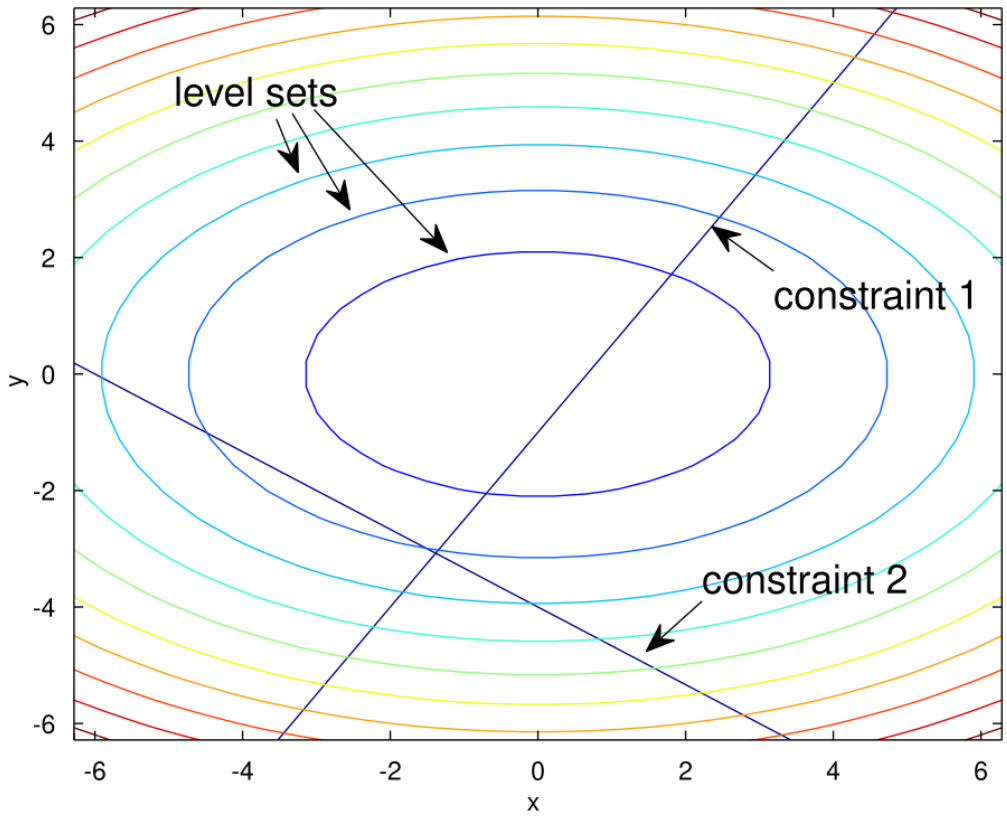


Menti-Quiz (Code: 2875-9119)

Why is the constraint enforced?

- At the maximum point $\nabla_\alpha \mathcal{L}(x, y, \alpha) = 0$
- This means $g(x, y) = c$.

Lagrangian with Multiple Constraints



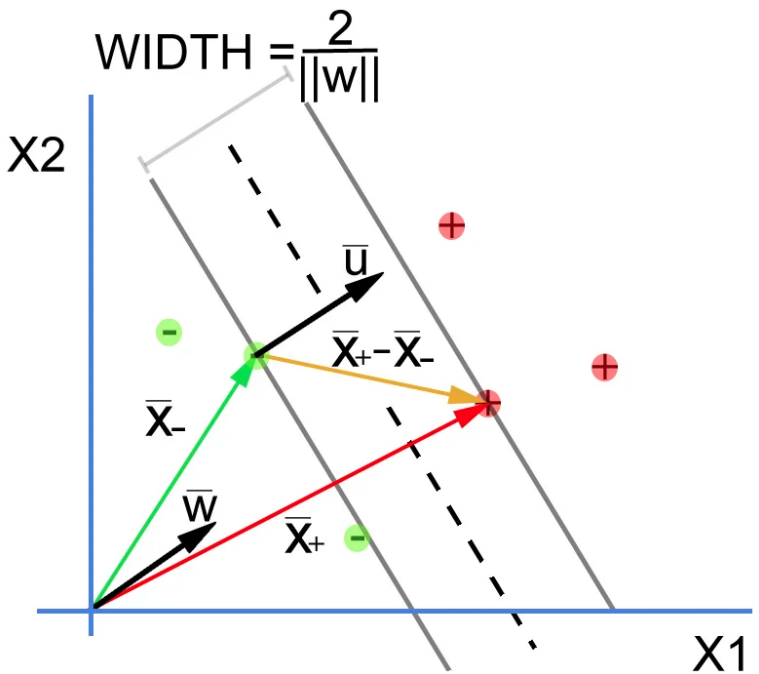
- Nothing restricts us to only one constraint.
- The framework allows for multiple **constraints**.
- We need this for the SVM margin objective.

Lagrangian Dual of the SVM objective

- Constraints become a part of the objective function as a **linear combination**.
- Involves additional variables - **Lagrange multipliers**.

$$\mathcal{L} = \underset{\mathbf{w}}{\text{argmin}}\ \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{n=1}^{N}\alpha_n(t_n(\mathbf{w}^T\mathbf{x}_n + b) - 1)$$

Subject to: $\alpha_n \geq 0$



Computing the derivatives

Dual SVM objective

$$\mathcal{L} = \text{argmin } \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{n=1}^{N} \alpha_n (t_n(\mathbf{w}^T\mathbf{x}_n + b) - 1)$$

- We know that $\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 0$ and $\frac{\partial \mathcal{L}}{\partial b} = 0$.

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \underbrace{\mathbf{w}}_{\text{Quadratic to linear}} - \sum_n \underbrace{\alpha_n t_n \mathbf{x}_n}_{\text{linear terms vanish}} = 0$$

$$\frac{\partial \mathcal{L}}{\partial b} = \underbrace{0}_{\mathbf{w} \text{ constant w.r.t. } b} - \sum_n \underbrace{\alpha_n t_n}_{\mathbf{w}^T\mathbf{x} \text{ vanish}} = 0$$

- Rearrange to get:

$$\mathbf{w} = \sum_n \alpha_n t_n \mathbf{x}_n$$

$$\sum_n \alpha_n t_n = 0$$

Substitute

$\mathbf{w} = \sum_n \alpha_n t_n \mathbf{x}_n, \ \sum_n \alpha_n t_n = 0$

$$\mathcal{L} = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_n \alpha_n (t_n(\mathbf{w}^T\mathbf{x}_n + b) - 1)$$

$$\vdots \qquad \vdots \qquad \vdots$$

$$\mathcal{L} = \underbrace{\sum_n \alpha_n}_{\text{WHY?}} - \frac{1}{2}\underbrace{\sum_{n,m} \alpha_n \alpha_m t_n t_m \mathbf{x}_n^T \mathbf{x}_m}_{\text{WHY?}}$$

Menti-Quiz (Code: 2875-9119)

Why are the two components in this form?

Substitute

$\mathbf{w} = \sum_n \alpha_n t_n \mathbf{x}_n, \ \sum_n \alpha_n t_n = 0$

$$\begin{aligned}
\mathcal{L} &= \frac{1}{2}\mathbf{w}^\mathsf{T}\mathbf{w} - \sum_n \alpha_n (t_n(\mathbf{w}^\mathsf{T}\mathbf{x}_n + b) - 1) \\
&\vdots \qquad\qquad\qquad\qquad \vdots \qquad\qquad\qquad\qquad \vdots \\
\mathcal{L} &= \underbrace{\sum_n \alpha_n}_{\text{WHY?}} - \underbrace{\frac{1}{2}\sum_{n,m} \alpha_n \alpha_m t_n t_m \mathbf{x}_n^\mathsf{T}\mathbf{x}_m}_{\text{WHY?}}
\end{aligned}$$

Menti-Quiz (Code: 2875-9119)

Why are the two components in this form?

- **Quadratic terms** ($\mathbf{x}_i\mathbf{x}_i$) appear from the norm $\mathbf{w}^\mathsf{T}\mathbf{w}$.
- **Cross-quadratic terms** ($\mathbf{x}_i\mathbf{x}_j$) appear as $\mathbf{w}$ is now a linear combination of $\mathbf{x}_i$.
- $\sum_n \alpha_n t_n$ come from the b component.

SVM Optimization Problem

- This is the **Lagrangian dual objective** which is to be **maximized**.
- Constraints: $\alpha_n \geq 0, \quad \sum_n \alpha_n t_n = 0$

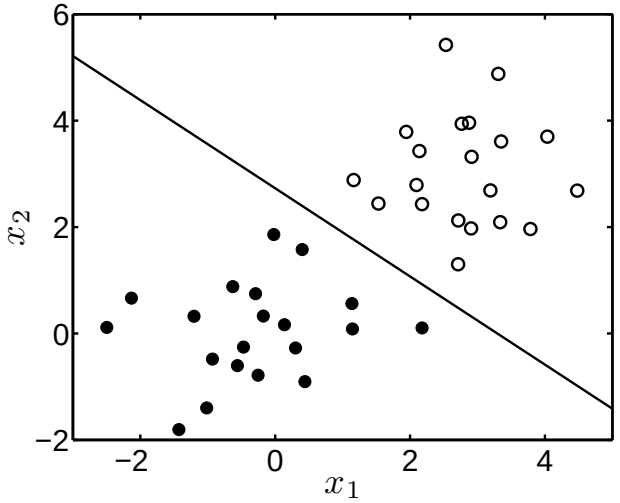
Optimization Problem

$$\underset{\boldsymbol{\alpha}}{\text{argmax}} \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n,m=1}^{N} \alpha_n \alpha_m t_n t_m \mathbf{x}_n^{\mathsf{T}} \mathbf{x}_m$$

$$\text{subject to} \quad \sum_{n=1}^{N} \alpha_n t_n = 0, \quad \alpha_n \geq 0$$

- Standard optimisation problem (**quadratic programming**)
- Has a **single**, **global** solution.
- Standard solver implementations.

Visualizing the decision boundary



- Optimizer outputs: $\alpha_1, \ldots, \alpha_N$
- Compute $\mathbf{w} = \sum_n \alpha_n t_n \mathbf{x}_n$
- Compute $b = t_n - \mathbf{w}^T\mathbf{x}$ (for one of the closest points)
  - $\mathbf{w}^T\mathbf{x} + b = \pm 1 = t_n$ for closest points.

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

$$\text{sign}\left(\sum_n \alpha_n t_n \mathbf{x}_n^T \mathbf{x}_{\text{new}} + b\right)$$

Introduction

D. Ganguly

Support Vector Machine

Margin

Maximising the margin

The Support Vector Machine

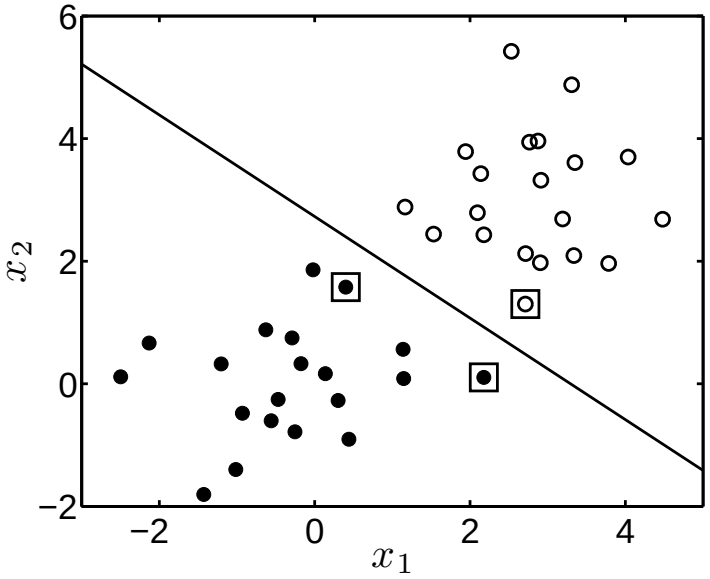
Soft margins

Kernels

Summary

Support Vectors (Result of Sparse Solutions)

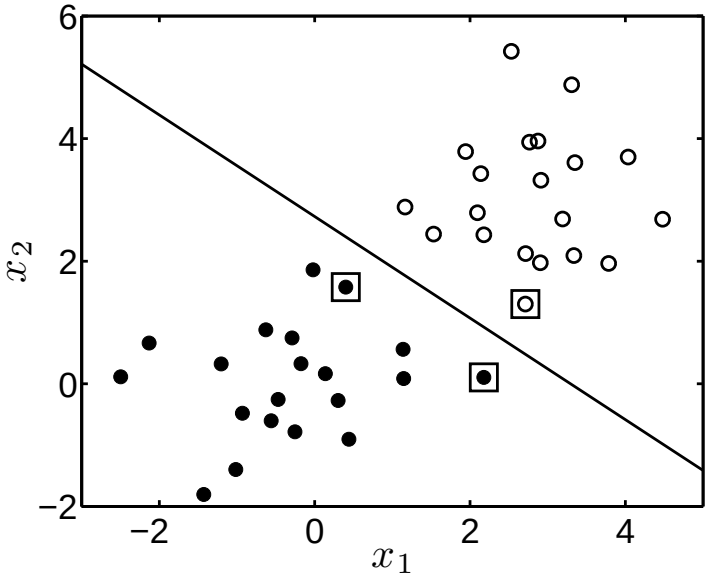
- At the optimum, only 3 non-zero α values (squares).



- $t_{\text{new}} = \text{sign}\left(\sum_n \alpha_n t_n \mathbf{x}_n^{\mathsf{T}} \mathbf{x}_{\text{new}} + b\right)$
- Predictions only depend on these data-points!
- Conforms to the geometric intuition: margin is only a function of closest points.

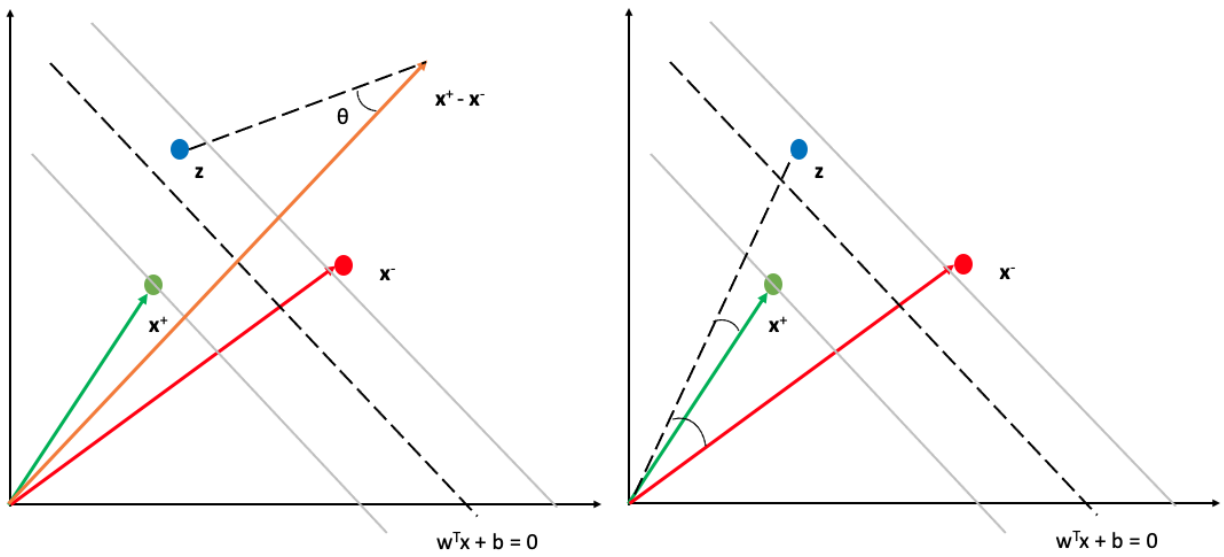
Support Vectors (Result of Sparse Solutions)

- At the optimum, only 3 non-zero α values (squares).



- $t_{\text{new}} = \text{sign}\left(\sum_n \alpha_n t_n \mathbf{x}_n^{\mathsf{T}} \mathbf{x}_{\text{new}} + b\right)$
- Predictions only depend on these data-points!
- Conforms to the geometric intuition: margin is only a function of closest points.
- These are called **Support Vectors**
- Normally a small proportion of the data:
  - Solution is *sparse*.

Faster Inference



- SVM Inference: $t(\mathbf{z}) = \text{sign}\left(\sum_n \alpha_i t_i \mathbf{x}_i \cdot \mathbf{z} + b\right)$, where $\mathbf{z}$ is a new point.
- We can precompute: $\mathbf{x} = \sum_i \alpha_i t_i \mathbf{x}_i$, and then during inference compute $t(\mathbf{z}) = \text{sign}\left(\mathbf{x} \cdot \mathbf{z} + b\right)$.
- This works because dot product distributes over vector addition, i.e., $(\mathbf{a} + \mathbf{b}) \cdot \mathbf{c} = \mathbf{a} \cdot \mathbf{c} + \mathbf{a} \cdot \mathbf{c}$.

Multi-class classification with SVMs?

Menti-Quiz (Code: 2875-9119)

We have seen that $t_n \pm 1$. How to model target variables for multiple classes?

Support Vector Machine

Margin

Maximising the margin

The Support Vector Machine

Soft margins

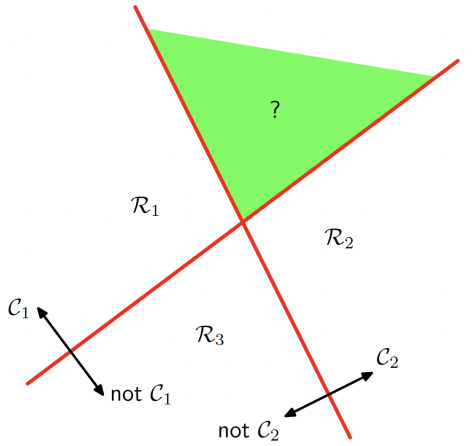
Kernels

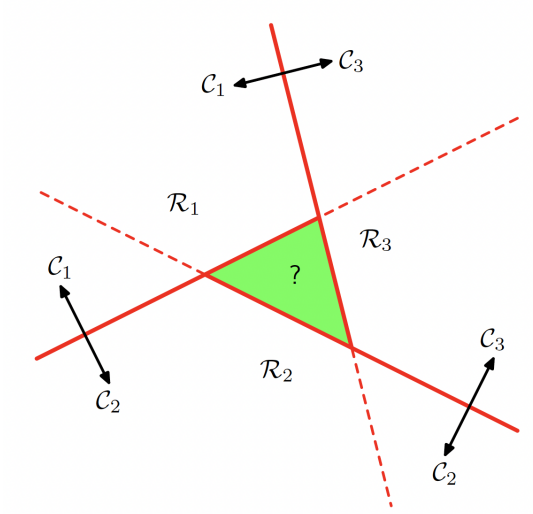
Summary

Multi-class classification with SVMs?

Menti-Quiz (Code: 2875-9119)

We have seen that $t_n \pm 1$. How to model target variables for multiple classes?





- One vs. rest boundaries.
- If for multiple classes $\mathbf{w}^T\mathbf{x} + b > 1$ take the one for which this value is maximum.
- Is this good?

- Pairwise boundaries.
- Aggregate the votes $((\mathbf{w}_i^T\mathbf{x} + b_i) - (\mathbf{w}_j^T\mathbf{x} + b_j))$ for final classification.
- Is this good?

Introduction

D. Ganguly

Support Vector Machine

Margin

Maximising the margin

The Support Vector Machine

Soft margins

Kernels

Summary

Pros and Cons of SVMs

Pros

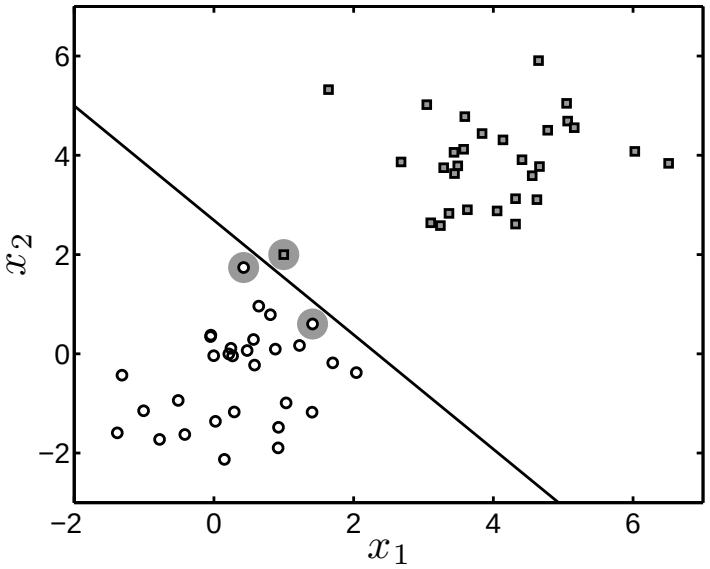
- Uncertainty doesn't depend on post-processing any more (as in Bayesian).
- Uncertainty modeled as a part of the objective function itself.
- Leads to sparse solutions (faster and better predictions).
- Less chance of overfitting on data (Why?)

Cons

- Not a probabilistic classifier
- Can we interpret probabilities as distance from the margin?
- How do we convert $\mathbf{w}^T\mathbf{x}_n + b$ values to probabilities?

How to regularise SVMs?

- Consider the decision boundary below.



- Decision boundary gets affected by an **outlier**.

$$t_n(\mathbf{w}^T\mathbf{x}_n + b) \geq 1$$

- Should constraints be so **hard**?
- Logistic regression:
  - **Penalization term** added to the objective - $||\theta||$ (norm of the parameter vector which we didn't want too high).
- What to do here?

Soft margin

- We can relax the constraints:

$$t_n(\mathbf{w}^T\mathbf{x}_n + b) \geq 1 - \xi_n, \qquad \underbrace{\xi_n}_{\text{slack variables}} \geq 0$$

- Primal Optimisation problem becomes:

$$\underset{\mathbf{w}}{\text{argmin}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + \underbrace{C\sum_{n=1}^{N}\xi_n}_{\text{We don't want too much of slack!}}$$

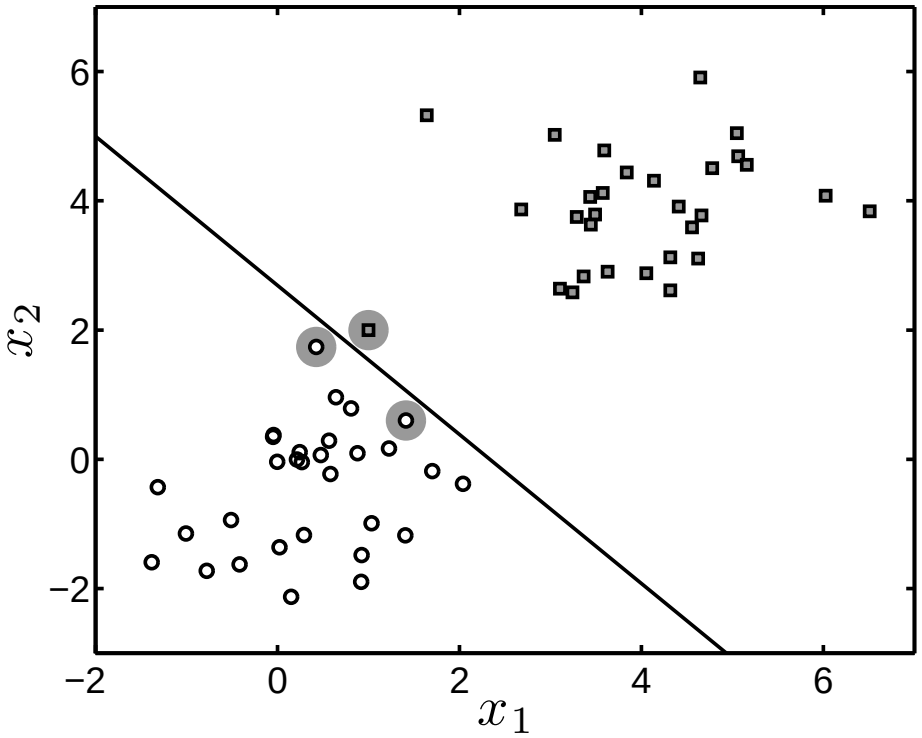
$$\text{subject to} \quad t_n(\mathbf{w}^T\mathbf{x}_n + b) \geq 1 - \xi_n$$

- Lagrangian Dual:

$$\underset{\boldsymbol{\alpha}}{\text{argmax}} \sum_{n=1}^{N}\alpha_n - \frac{1}{2}\sum_{n.m=1}^{N}\alpha_n\alpha_m t_n t_m \mathbf{x}_n^T\mathbf{x}_m$$

$$\text{subject to} \quad \sum_{n=1}^{N}\alpha_n t_n = 0, \quad \underbrace{0 \leq \alpha_n \leq C}_{\text{only change}}$$

Soft margins

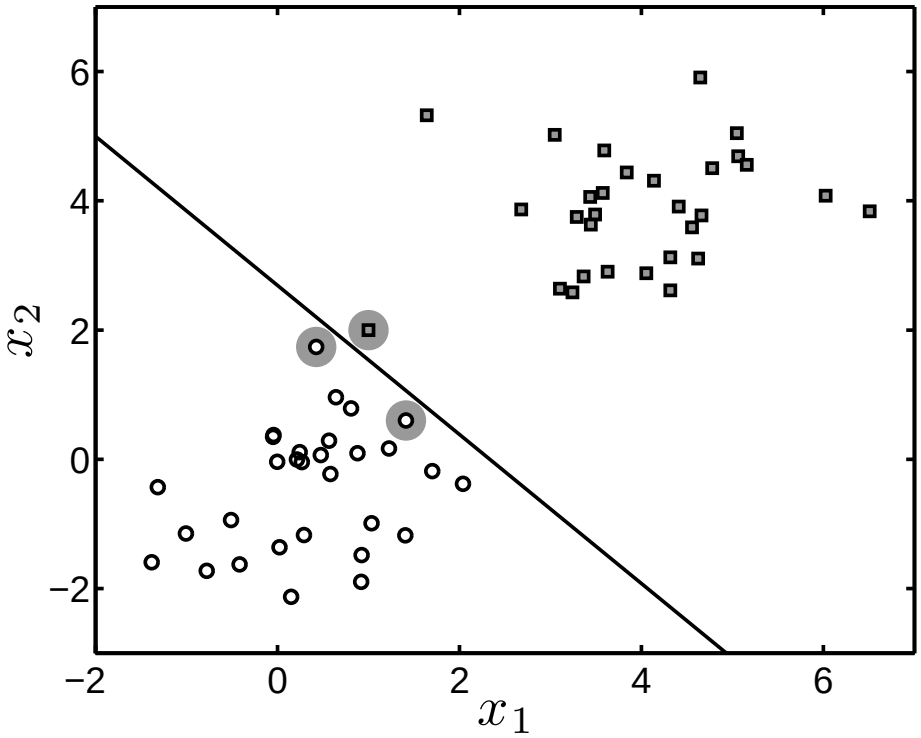


- α_n for the 'bad' square is 3.5 (we want to make this close to 0).

Menti-Quiz (Code: 2875-9119)

Why do we want to make α for the outlier close to 0?

Soft margins



- α_n for the 'bad' square is 3.5 (we want to make this close to 0).

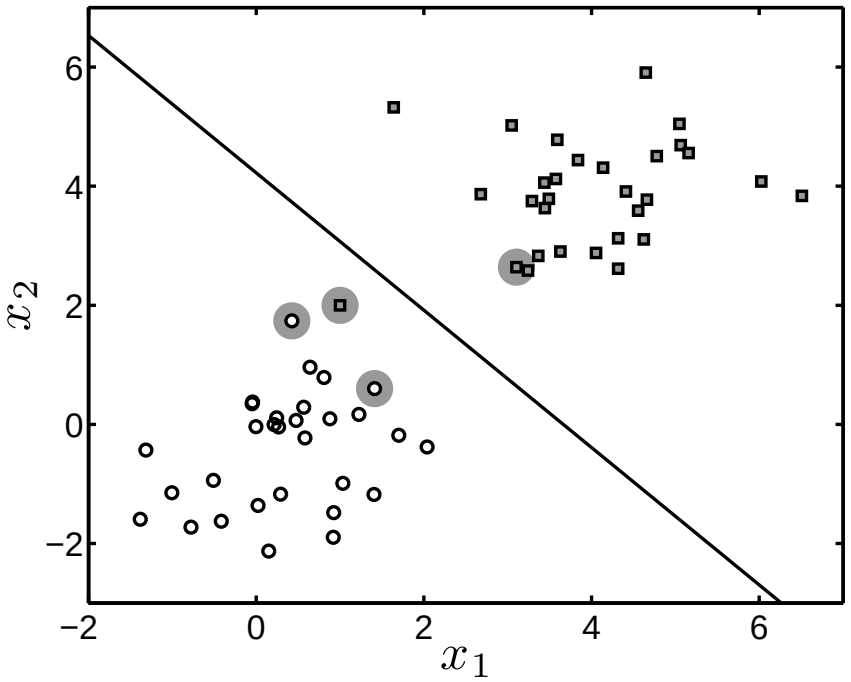
Menti-Quiz (Code: 2875-9119)

Why do we want to make α for the outlier close to 0?

Because we don't want this to be a support vector.

Soft margins

- Try $C = 1$



- We have an extra support vector.
- And a better decision boundary.

Soft margins

- The choice of C is very important. Typically chosen with cross-validation.

Menti-Quiz (Code: 2875-9119)

Too high and we **over-fit** to the data. Why?

Soft margins

- The choice of C is very important. Typically chosen with cross-validation.

Menti-Quiz (Code: 2875-9119)

Too high and we **over-fit** to the data. Why?

Menti-Quiz (Code: 2875-9119)

Too low and we **underfit**. Why?

Soft margins

- The choice of C is very important. Typically chosen with cross-validation.

Menti-Quiz (Code: 2875-9119)

Too high and we **over-fit** to the data. Why?

Menti-Quiz (Code: 2875-9119)

Too low and we **underfit**. Why?

- When no slack variables (i.e., when just $\alpha_i \geq 0$), C is essentially a very large number.
- Less variance between α_is $\rightarrow$ solution loses sparsity.

Some key points

- In our example, we started with 3 parameters:

$$\mathbf{w} = [w_1, w_2]^{\mathsf{T}}, \quad b$$

- In general: $D+1$ (D: data dimension).
- We now have N: $\alpha_1, \ldots, \alpha_N$ (N: number of training data points)
  - Is this a harder problem than gradient descent on $D+1$ dimensions?
- Worth the effort because:
  - We model data uncertainty (maximizing the confidence of our predictions).
  - w/o computationally expensive (and often intractable for high dimensions) Bayesian approaches!

Inner products in the SVM decision boundary

- Optimisation problem:

$$\underset{\boldsymbol{\alpha}}{\text{argmax}} \sum_n \alpha_n - \frac{1}{2} \sum_{n,m} \alpha_n \alpha_m t_n t_m \mathbf{x}_n^T \mathbf{x}_m$$

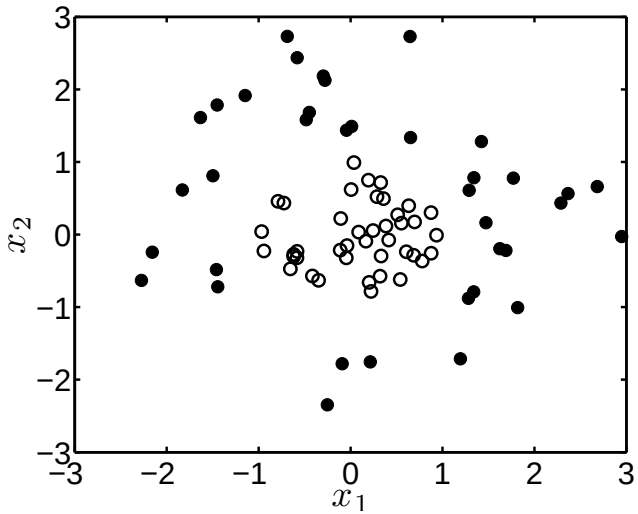
- Decision function:

$$t_{\text{new}} = \text{sign}\left(\sum_n \alpha_n t_n \mathbf{x}_n^T \mathbf{x}_{\text{new}} + b \right)$$

- Data instances appear only as **inner products**.

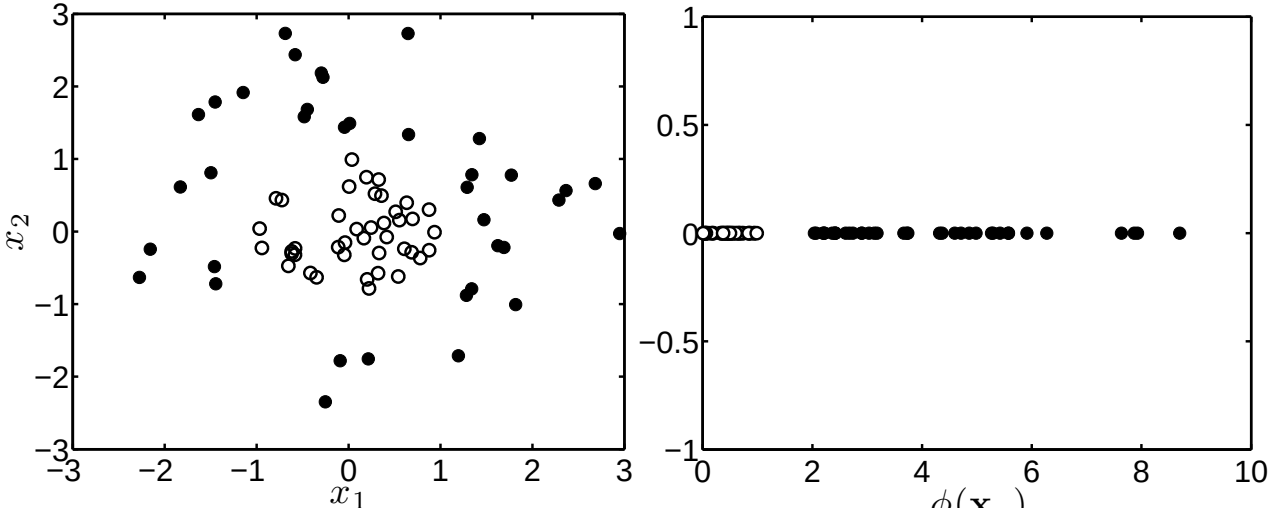
Projections

- Our SVM can find linear decision boundaries.
- What if the data requires something nonlinear?



Projections

- Our SVM can find linear decision boundaries.
- What if the data requires something nonlinear?



- We can transform the data e.g.:

$$\phi(\mathbf{x}_n) = x_{n1}^2 + x_{n2}^2$$

- So that it can be separated with a straight line.
- And use $\phi(\mathbf{x}_n)$ instead of $\mathbf{x}_n$ in our optimisation.

Projections

- Our optimisation is now:

$$\underset{\boldsymbol{\alpha}}{\text{argmax}} \sum_n \alpha_n - \frac{1}{2} \sum_{n,m} \alpha_n \alpha_m t_n t_m \phi(\mathbf{x}_n)^\mathsf{T} \phi(\mathbf{x}_m)$$

- And predictions:

$$t_{\text{new}} = \text{sign}\left(\sum_n \alpha_n t_n \phi(\mathbf{x}_n)^\mathsf{T} \phi(\mathbf{x}_{\text{new}}) + b \right)$$

Projections

- Our optimisation is now:

$$\underset{\boldsymbol{\alpha}}{\text{argmax}} \sum_n \alpha_n - \frac{1}{2} \sum_{n,m} \alpha_n \alpha_m t_n t_m \phi(\mathbf{x}_n)^\mathsf{T} \phi(\mathbf{x}_m)$$

- And predictions:

$$t_{\text{new}} = \text{sign}\left(\sum_n \alpha_n t_n \phi(\mathbf{x}_n)^\mathsf{T} \phi(\mathbf{x}_{\text{new}}) + b \right)$$

- In this case:

$$\phi(\mathbf{x}_n^\mathsf{T})\phi(\mathbf{x}_m) = (x_{n1}^2 + x_{n2}^2)(x_{m1}^2 + x_{m2}^2) = k(\mathbf{x}_n, \mathbf{x}_m)$$

Projections

- Our optimisation is now:

$$\underset{\boldsymbol{\alpha}}{\text{argmax}} \sum_n \alpha_n - \frac{1}{2} \sum_{n,m} \alpha_n \alpha_m t_n t_m \phi(\mathbf{x}_n)^\mathsf{T} \phi(\mathbf{x}_m)$$

- And predictions:

$$t_{\text{new}} = \text{sign}\left(\sum_n \alpha_n t_n \phi(\mathbf{x}_n)^\mathsf{T} \phi(\mathbf{x}_{\text{new}}) + b \right)$$

- In this case:

$$\phi(\mathbf{x}_n^\mathsf{T})\phi(\mathbf{x}_m) = (x_{n1}^2 + x_{n2}^2)(x_{m1}^2 + x_{m2}^2) = k(\mathbf{x}_n, \mathbf{x}_m)$$

- We can think of the dot product in the projected space as a function of the original data.

Projections

- No need to separately compute the projections (different from feature maps).
- Can just think of functions $k(\mathbf{x}_n, \mathbf{x}_m)$ that are **dot products in some space**.
  - Called **kernel** functions.
- Useful because we can directly define the kernel function k w/o even defining ϕ.
- A natural interpretation of k: Defines an abstract notion of **task-specific similarity**.

Projections

- No need to separately compute the projections (different from feature maps).
- Can just think of functions $k(\mathbf{x}_n, \mathbf{x}_m)$ that are **dot products in some space**.
  - Called **kernel** functions.
- Useful because we can directly define the kernel function k w/o even defining ϕ.
- A natural interpretation of k: Defines an abstract notion of **task-specific similarity**.
- Optimisation task:

$$\underset{\boldsymbol{\alpha}}{\text{argmax}} \sum_n \alpha_n - \frac{1}{2} \sum_{n,m} \alpha_n \alpha_m t_n t_m \, k(\mathbf{x}_n, \mathbf{x}_m)$$

- Predictions:

$$t_{\text{new}} = \text{sign}\left(\sum_n \alpha_n t_n \, k(\mathbf{x}_n, \mathbf{x}_{\text{new}}) + b \right)$$

Off-the-shelf Kernels

- Linear (Conventional similarity):

$$k(\mathbf{x}_n, \mathbf{x}_m) = \mathbf{x}_n^{\mathsf{T}} \mathbf{x}_m$$

- Gaussian (Data equidistant from a center to be considered more similar):

$$k(\mathbf{x}_n, \mathbf{x}_m) = \exp\left\{-\beta(\mathbf{x}_n - \mathbf{x}_m)^{\mathsf{T}}(\mathbf{x}_n - \mathbf{x}_m)\right\}$$

- Polynomial (similarities with higher order features):

$$k(\mathbf{x}_n, \mathbf{x}_m) = (1 + \mathbf{x}_n^{\mathsf{T}} \mathbf{x}_m)^{\beta}$$

Off-the-shelf Kernels

- Linear (Conventional similarity):

$$k(\mathbf{x}_n, \mathbf{x}_m) = \mathbf{x}_n^{\mathsf{T}} \mathbf{x}_m$$

- Gaussian (Data equidistant from a center to be considered more similar):

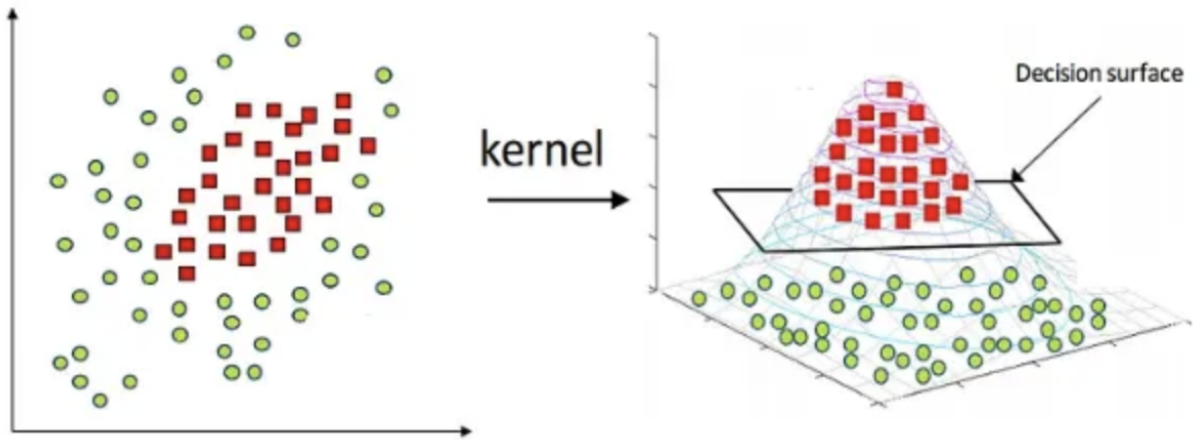
$$k(\mathbf{x}_n, \mathbf{x}_m) = \exp\left\{-\beta(\mathbf{x}_n - \mathbf{x}_m)^{\mathsf{T}}(\mathbf{x}_n - \mathbf{x}_m)\right\}$$

- Polynomial (similarities with higher order features):

$$k(\mathbf{x}_n, \mathbf{x}_m) = (1 + \mathbf{x}_n^{\mathsf{T}} \mathbf{x}_m)^{\beta}$$

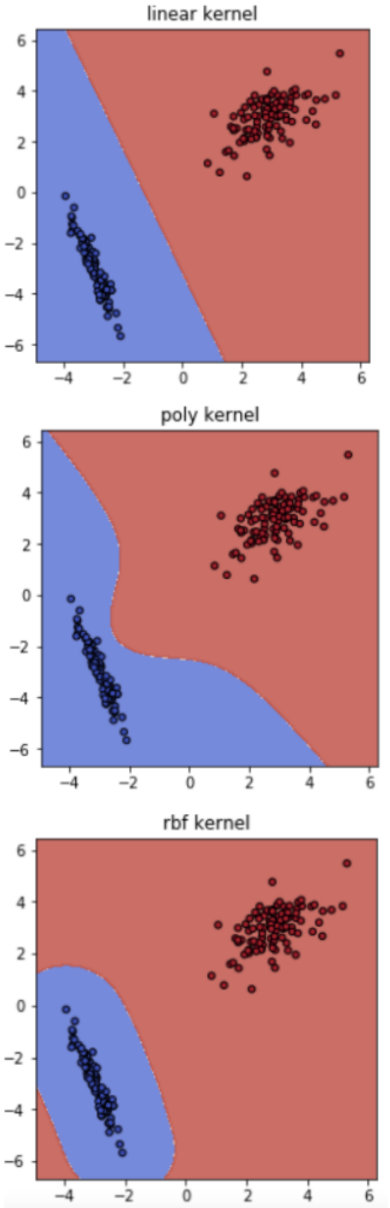
- These all correspond to $\phi(\mathbf{x}_n)^{\mathsf{T}}\phi(\mathbf{x}_m)$ for some transformation $\phi(\mathbf{x}_n)$.
- Don't know what the projections $\phi(\mathbf{x}_n)$ are – don't need to know!

Kernelized SVM



- Finds linear boundaries in a transformed (higher dimensional) space mapping back to non-linear boundaries in the original space.
  - Still a quadratic program with a single, global optimum.
- Finds complex decision boundaries with a linear algorithm
  - Analogous to feature maps for gradient-based approaches.

How to plot the boundary in Kernelized SVM?



- Our decision boundary was defined as $\mathbf{w}^T\mathbf{x} + b = 0$.
- $\mathbf{w} = \sum_{n=1}^{N} \alpha_n t_n \phi(\mathbf{x}_n)$
- As don't know $\phi(\mathbf{x}_n)$, we can't compute $\mathbf{w}$ or the boundary!
- But we can evaluate the predictions on a grid of points using the kernel function to draw a contour:

$$\sum_{n=1}^{N} \alpha_n t_n k(\mathbf{x}_n, \mathbf{x}_{\text{new}}) + b$$

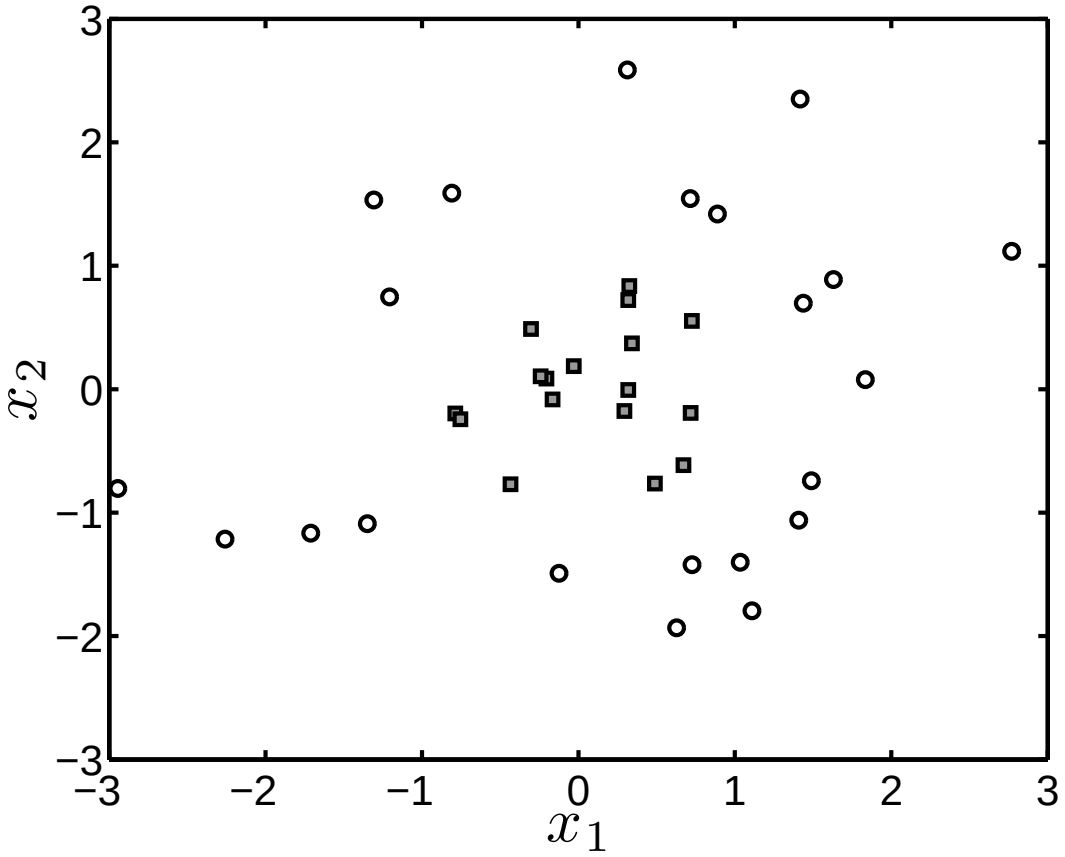
Kernelising other algorithms

- With the **kernel trick**, simple algorithms can be used to solve complex problems!
- Any algorithm where data appears as inner products can be kernelised, e.g., KNN.

Kernelized KNN

- Least distance neighborhood $\rightarrow$ Maximum inner product neighborhood.
- $N_k(\mathbf{x}_{\text{new}}) = \arg\max \kappa(\mathbf{x}_{\text{new}}, \mathbf{x}_i)$ (κ: kernel function)

Gaussian Kernel Example



- We'll use a Gaussian kernel:

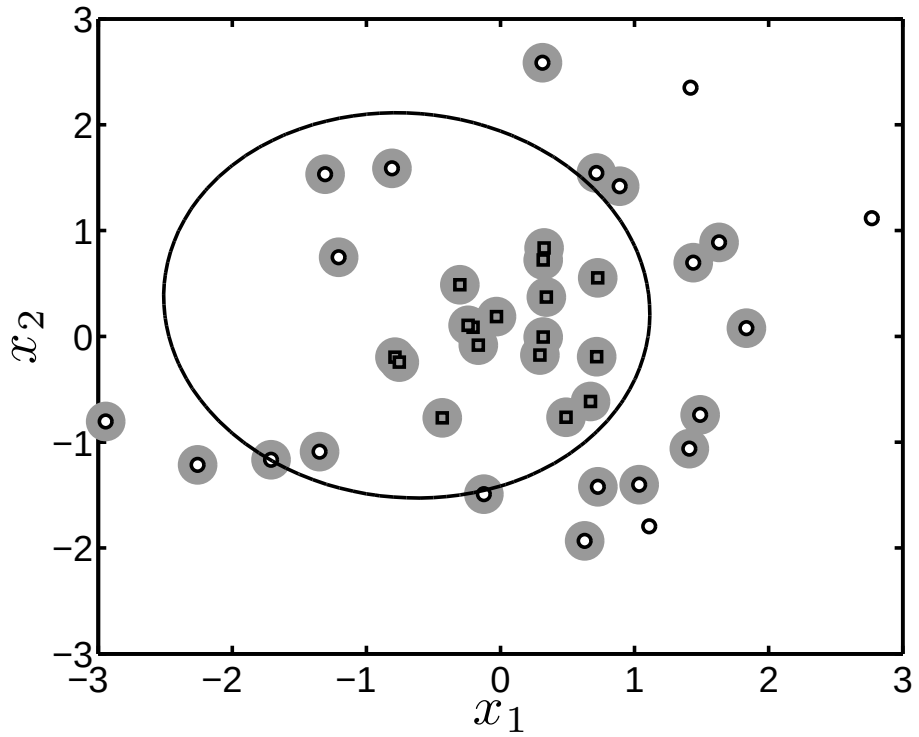
$$k(\mathbf{x}_n, \mathbf{x}_m) = \exp\left\{-\beta(\mathbf{x}_n - \mathbf{x}_m)^{\mathsf{T}}(\mathbf{x}_n - \mathbf{x}_m)\right\}$$

- And vary β ($C = 10$).

Gaussian Kernel Example

$$k(\mathbf{x}_n, \mathbf{x}_m) = \exp\left\{-\beta(\mathbf{x}_n - \mathbf{x}_m)^{\mathsf{T}}(\mathbf{x}_n - \mathbf{x}_m)\right\}$$

β controls the *complexity* of the decision boundaries.

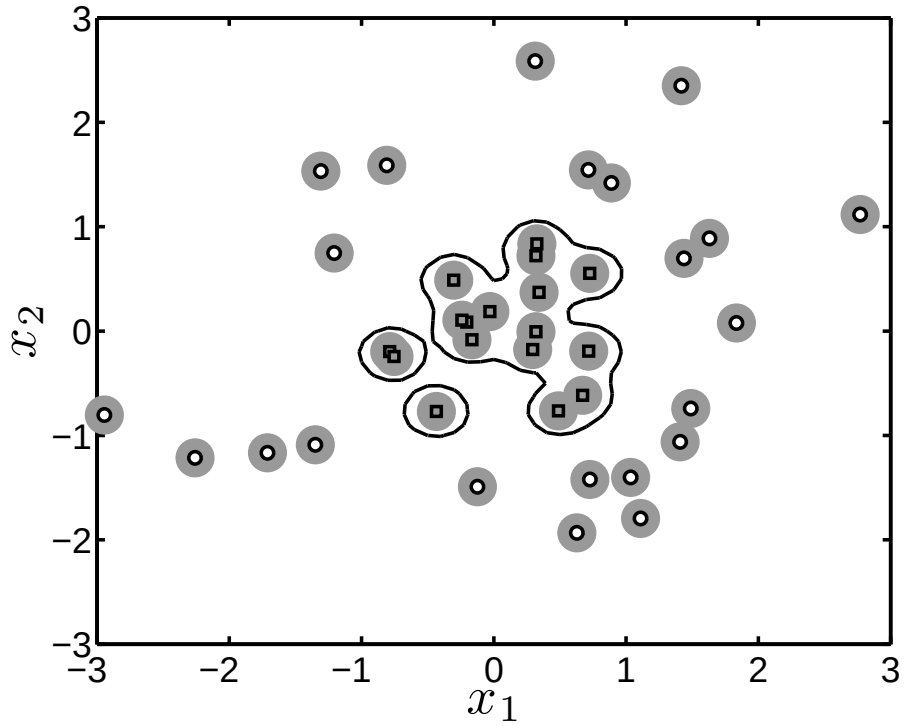


- $\beta = 0.01$
- Not flexible enough to surround just the square class.
- Non-sparse solution.

Gaussian Kernel Example

$$k(\mathbf{x}_n, \mathbf{x}_m) = \exp\left\{-\beta(\mathbf{x}_n - \mathbf{x}_m)^{\mathsf{T}}(\mathbf{x}_n - \mathbf{x}_m)\right\}$$

β controls the *complexity* of the decision boundaries.

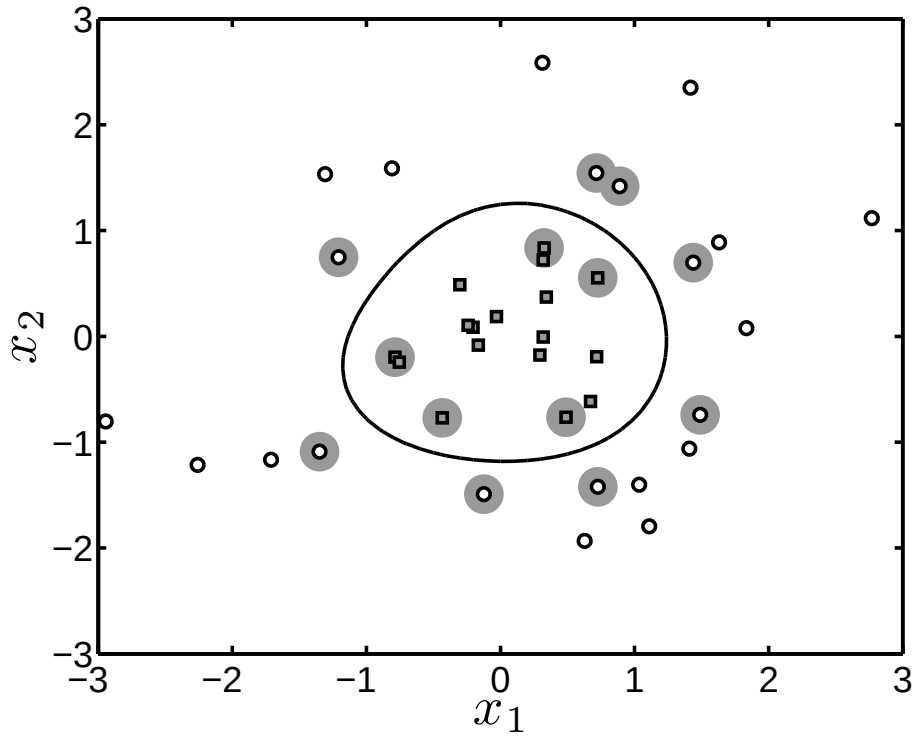


- $\beta = 50$
- Overfits on the data.
- Non-sparse solution.

Gaussian Kernel Example

$$k(\mathbf{x}_n, \mathbf{x}_m) = \exp\left\{-\beta(\mathbf{x}_n - \mathbf{x}_m)^{\mathsf{T}}(\mathbf{x}_n - \mathbf{x}_m)\right\}$$

β controls the *complexity* of the decision boundaries.



- $\beta = 1$
- Right amount of sparsity.
- Likely to generalise well.

Choosing kernel function, parameters and C

- Kernel function and parameter choice is data dependent.
- Easy to overfit.

Choosing kernel function, parameters and C

- Kernel function and parameter choice is data dependent.
- Easy to overfit.
- Need to set C too
- C and β are *linked*
  - C too high – overfitting.
  - C too low – underfitting.

Choosing kernel function, parameters and C

- Kernel function and parameter choice is data dependent.
- Easy to overfit.
- Need to set C too
- C and β are *linked*
  - C too high – overfitting.
  - C too low – underfitting.
- Cross-validation!

Choosing kernel function, parameters and C

- Kernel function and parameter choice is data dependent.
- Easy to overfit.
- Need to set C too
- C and β are *linked*
  - C too high – overfitting.
  - C too low – underfitting.
- Cross-validation!
- Search over β and C

Summary - SVMs

- Described a classifier that is optimised by **maximising the margin**.
- Did some **re-arranging** to turn it into a **quadratic programming problem**.
- **Loosened the SVM constraints** to allow points on the wrong side of boundary.
- Saw that data only **appear as inner products**.
- Introduced the **idea of kernels**.