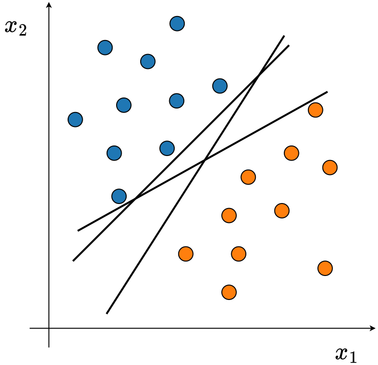
Support vector machines

Separating hyperplanes

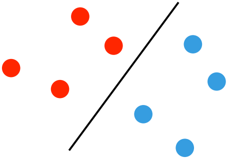
- **Goal:** Build hyperplanes that separate points in two classes,
- **Question:** Which is the best separating line?



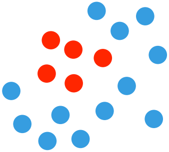
Non-linear separation

Non-linear separation

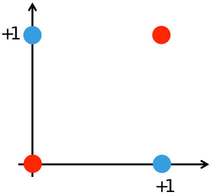
- **Question:** How to handle non-linearly separable data points?



Linearly separable



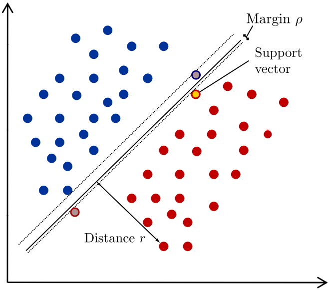
Nonlinearly separable



The xor function

Classification margin

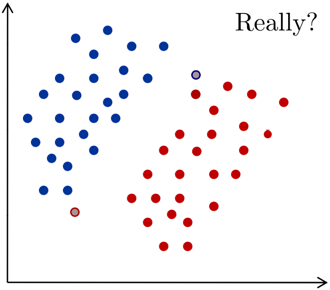
- Signed distance from an example to the separator: $r = \dfrac{\boldsymbol{w}^\mathsf{T}\boldsymbol{x} + b}{\|\boldsymbol{w}\|}$
- Examples closest to the hyperplane are support vectors.
- Margin ρ is the distance between the separator and the support vector(s).



- Is this one good?
- Consider two testing samples, where are they assigned?
- This separator may have large generalization error.

Classification margin

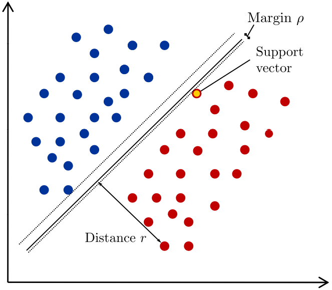
- Signed distance from an example to the separator: $r = \dfrac{\boldsymbol{w}^{\mathsf{T}}\boldsymbol{x} + b}{\|\boldsymbol{w}\|}$
- Examples closest to the hyperplane are support vectors.
- Margin ρ is the distance between the separator and the support vector(s).



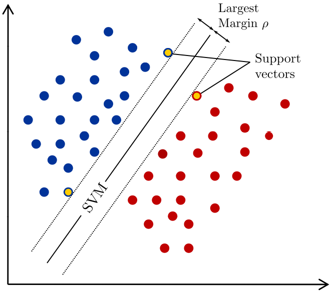
- Is this one good?
- Consider two testing samples, where are they assigned?
- This separator may have large generalization error.

Largest margin

- Maximizing the margin reduces generalization error
- Then, there are necessary support vectors on both sides of the hyperplane
- Only support vectors are important → other examples can be discarded



Poor separation



Optimal separation

Margin formulation

In SVM, the target vectors are $y_i = +1$ or $y_i = -1$.

We are looking for parameters $(\boldsymbol{w}, b)$ such that:

$$\boldsymbol{w}^\mathsf{T}\boldsymbol{x}_i + b \geqslant +1 \quad \text{if} \quad y_i = +1$$
$$\boldsymbol{w}^\mathsf{T}\boldsymbol{x}_i + b \leqslant -1 \quad \text{if} \quad y_i = -1$$

or in short: $\quad y_i(\boldsymbol{w}^\mathsf{T}\boldsymbol{x}_i + b) \geqslant +1 \quad \text{and} \quad |\boldsymbol{w}^\mathsf{T}\boldsymbol{x}_i + b| \geqslant 1$

For support vectors $\boldsymbol{x}_s$, this inequality can be forced to be an equality (parameters $\boldsymbol{w}$ and b can always be renormalized to satisfy this):

$$|\boldsymbol{w}^\mathsf{T}\boldsymbol{x}_s + b| = 1$$

Then the margin is given by:

$$\rho = |r| = \frac{|\boldsymbol{w}^\mathsf{T}\boldsymbol{x}_s + b|}{\|\boldsymbol{w}\|} = \frac{1}{\|\boldsymbol{w}\|}$$

Margin maximization problem

The margin maximization problem can be formulated as a minimization:

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 \quad \text{subject to} \quad y_i(\boldsymbol{w}^\mathsf{T}\boldsymbol{x}_i + b) \geqslant 1 \quad \forall\, i = 1, \ldots, N$$

This is a **quadratic** and **convex** optimization problem, subject to linear constraints $\rightarrow$ There exists numerical solvers for this type of problem.

We can reformulate the problem using Lagrange multipliers $\alpha_i \geq 0$:

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{N} \alpha_i \left(y_i(\boldsymbol{w}^\mathsf{T}\boldsymbol{x}_i + b) - 1\right)$$

After cancelling derivatives of the Lagrangian $L(\boldsymbol{w}, b, \boldsymbol{\alpha})$ with respect to $\boldsymbol{w}$ and b, we find:

$$\boldsymbol{w} = \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}_i \quad \text{and} \quad \sum_{i=1}^{N} \alpha_i y_i = 0$$

Dual representation

By plugging these into the Lagrangian, we obtain the following maximization problem (called the *dual problem*):

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j \, y_i y_j \, \boldsymbol{x}_i^\mathsf{T} \boldsymbol{x}_j$$

$$\text{subject to} \quad \alpha_i \geqslant 0 \quad \text{and} \quad \sum_i \alpha_i y_i = 0, \quad \forall i = 1, \ldots, N$$

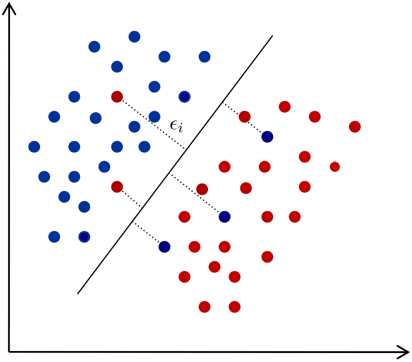
which can be solved with quadradic programming also.

It can be shown that **non-zero α_i correspond to support vectors $\boldsymbol{x}_i$** *(at test time, only support vectors are considered)*.

- In practice, the dual problem is more efficient to solve, and allows identifying the support vectors
- It only depends on the dot products $\boldsymbol{x}_i^\mathsf{T} \boldsymbol{x}_j$ between training points

Non linearly separable data

What if the training set is not linearly separable?



The optimization problem does not admit solutions.

Solution: allow the model to make errors ϵ_i for some $\boldsymbol{x}_i$.

Soft margin SVM

- Relax the constraints by permitting errors to some extent:

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_i \max(0, 1 - y_i(\boldsymbol{w}^\mathsf{T}\boldsymbol{x}_i + b))$$

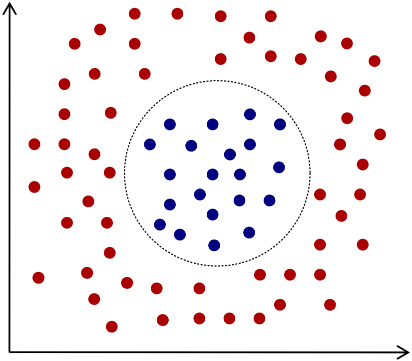
- The new term is called the **hinge loss**
- The hyperparameter $C > 0$ controls overfitting
- The dual problem becomes

$$\max_{\boldsymbol{\alpha}} \sum_i \alpha_i - \frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j\, y_i y_j\, \boldsymbol{x}_i^\mathsf{T}\boldsymbol{x}_j$$

subject to $0 \leqslant \alpha_i \leqslant C$ and $\sum_i \alpha_i y_i = 0,$ for all i.

- It is still a convex quadradic optimization problem.

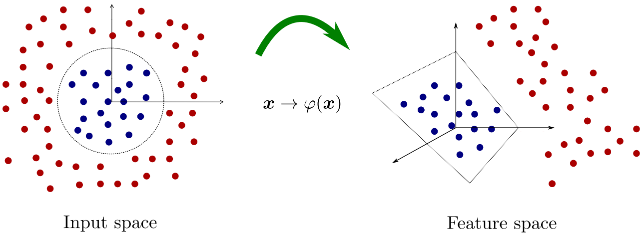
Non-linear SVM – Overview



How to obtain non-linear decision boundaries?

Higher dimension feature maps

Map features to higher dimensions



Here: $(x_1, x_2) \mapsto (x_1^2, x_2^2, \sqrt{2}x_1x_2)$

The input space (original feature space) can always be mapped to some higher-dimensional feature space where the training data set is linearly separable, via some (non-linear) transformation $\boldsymbol{x} \to \phi(\boldsymbol{x})$.

Kernel trick

- The mapping $\phi(\boldsymbol{x})$ may be **very difficult** to find and may require **exponentially more** parameters to learn
- However, in its dual formulation, SVM does not rely on the mapping itself, but only on the dot products between them $\phi(\boldsymbol{x})^\mathsf{T}\phi(\boldsymbol{x}')$
- **The trick**: there exists[1] **kernel functions** such that

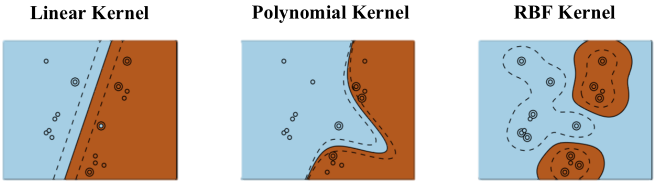
$$K(\boldsymbol{x}, \boldsymbol{x}') = \phi(\boldsymbol{x})^\mathsf{T}\phi(\boldsymbol{x}')$$

- The kernel function $K(\boldsymbol{x}, \boldsymbol{x}')$ can be used as a replacement, without knowing the actual mapping $\phi(\boldsymbol{x})$
- Generic tool: can also be applied to other algorithms (linear regression, logistic regression, etc.)

[1] See Mercer's theorem

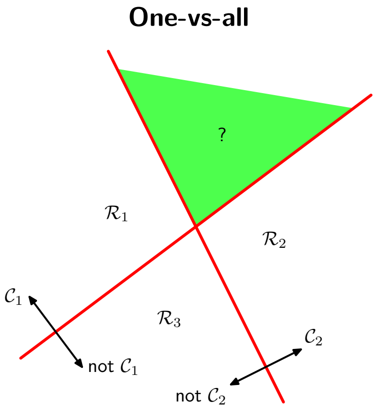
Examples of kernels

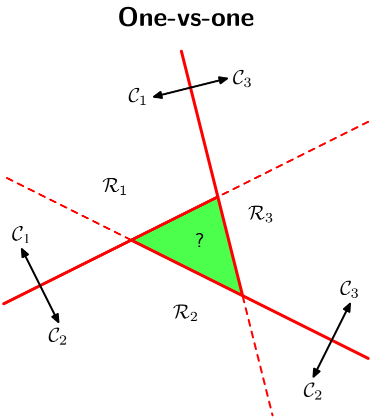
- Linear: $K(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^\top \boldsymbol{x}'$
- Polynomial: $K(\boldsymbol{x}, \boldsymbol{x}') = (\gamma \boldsymbol{x}^\top \boldsymbol{x}' + \beta)^p$
- Gaussian (RBF): $K(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\gamma \|\boldsymbol{x} - \boldsymbol{x}'\|^2\right)$
- Sigmoid: $K(\boldsymbol{x}, \boldsymbol{x}') = \tanh(\gamma \boldsymbol{x}^\top \boldsymbol{x}' + \beta)$



In practice, K is usually chosen with a validation set $\mathcal{D}_{\text{valid}}$

Multiclass setting with K classes





- Indecision region: weight votes by probabilities $p(\mathcal{C}_k \mid \boldsymbol{x})$