Support Vector Machine

Machine Learning Techniques

Dr. Ashish Tendulkar
IIT Madras

We will learn **support vector machines** (SVM) in this module.

Overview

- SVM is a supervised machine learning algorithm that can be used for both classification and regression problems.

- We will focus on classification aspect of SVM in our course.

- SVM is a discriminative classifier like perceptron and logistic regression

- SVM works in both binary and multiclass classification set ups.

Following our template, we will describe all five components of ML set up for SVM one by one.

Training data is the first component of our ML framework.

Training data

- In binary classification set up, training data consists of
  - Feature matrix $\mathbf{X}$ with shape (n, m). Note that each example is represented with m features and there are total n examples.
  - Label vector $\mathbf{y}$ containing labels for n examples and its shape is $(n,)$.

- In multiclass and multilabel set ups, training data consists of feature matrix with the same specification as the binary set up.
  - Label matrix $\mathbf{Y}$ containing one of k labels for each of n examples and its shape is (n, k).

Model is the second component of our ML framework, which will be discussed in the context of a binary classification set up.

Model

Weight vector

$$y = \mathrm{sign}(\mathbf{w}^T\mathbf{x} + b)$$

Label

Feature vector

Linear combination
of features

- Note that we have written bias unit separately over here. This is a linear classifier, which is familiar to us.
- Labels are assumed to be +1 and -1.

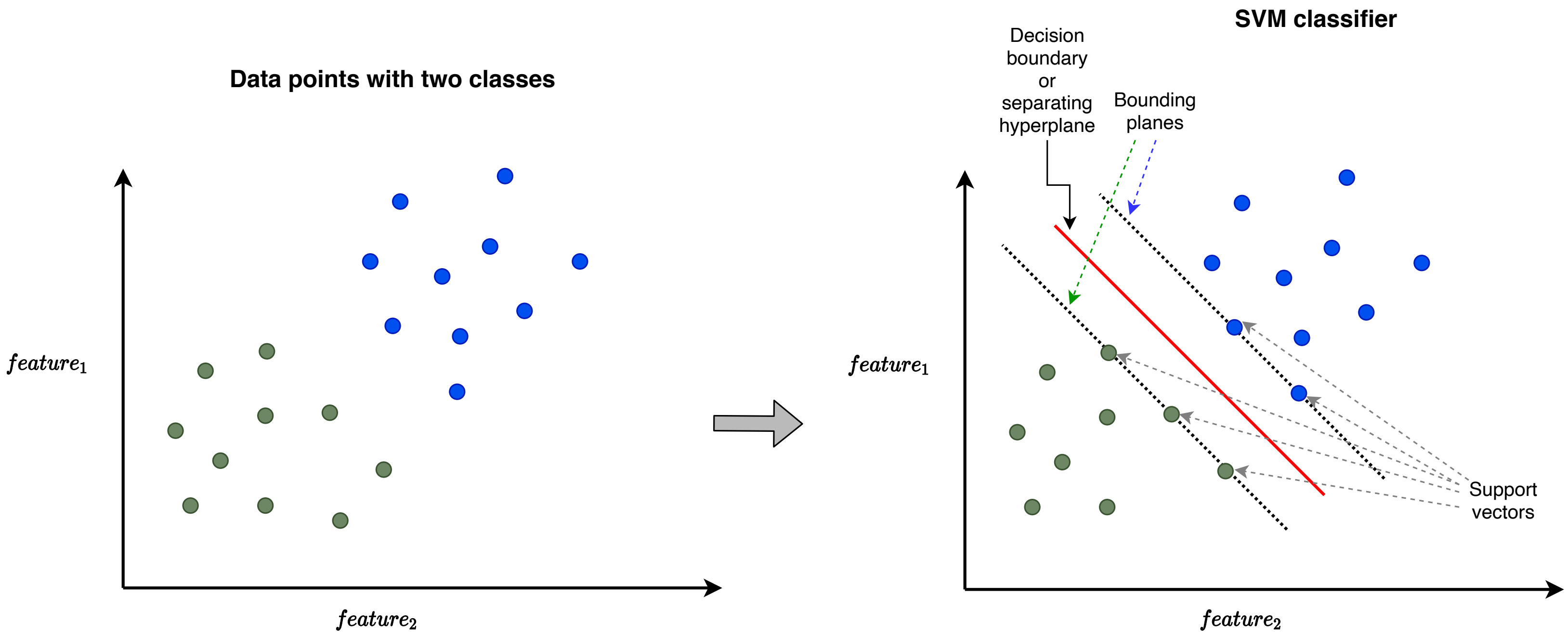
Learning problem

$$y = \mathrm{sign}(\mathbf{w}^T\mathbf{x} + b)$$

The SVM learns a hyperplane separating two classes with parameters $\mathbf{w}$ and b.

Given the training data, find the best values for $\mathbf{w}$ and b that results into the minium loss.

SVM finds the hyperplane in slightly different manner that other classifiers that we have studied in this course.



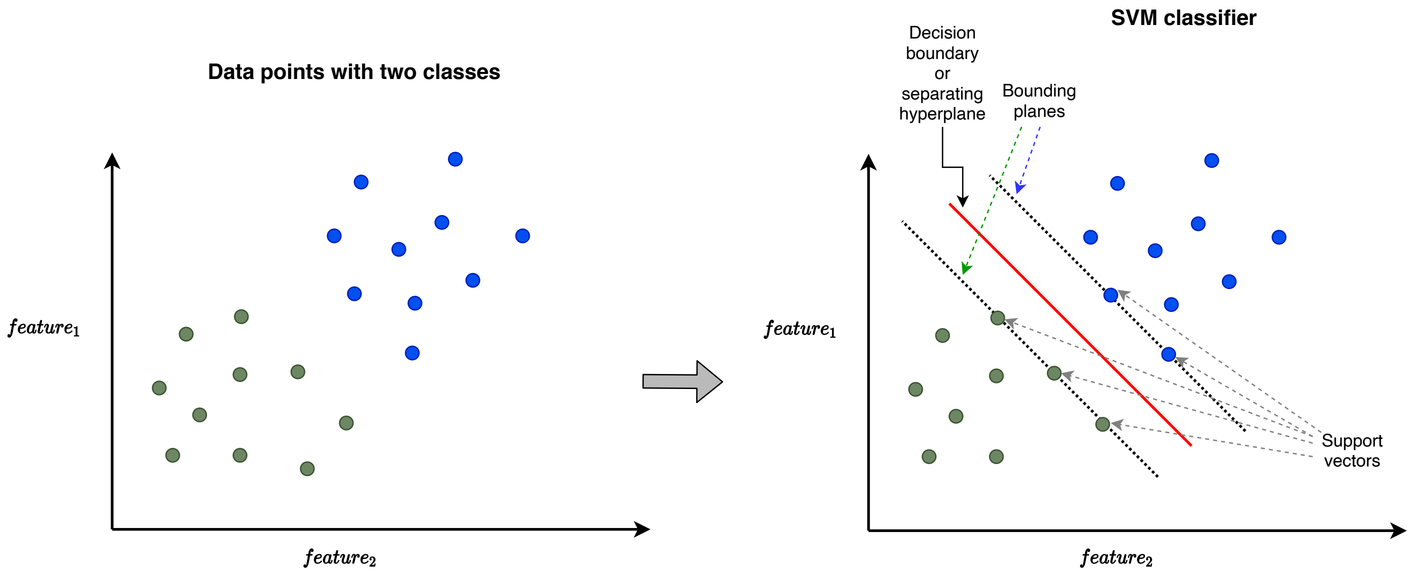
It selects the hyperplane that maximizes the distance to the closest data points from both classes.

In other words, it is the hyperplane with maximum margin between two classes.

How does it select such a hyperplane? It uses appropriate loss functions for the objective.

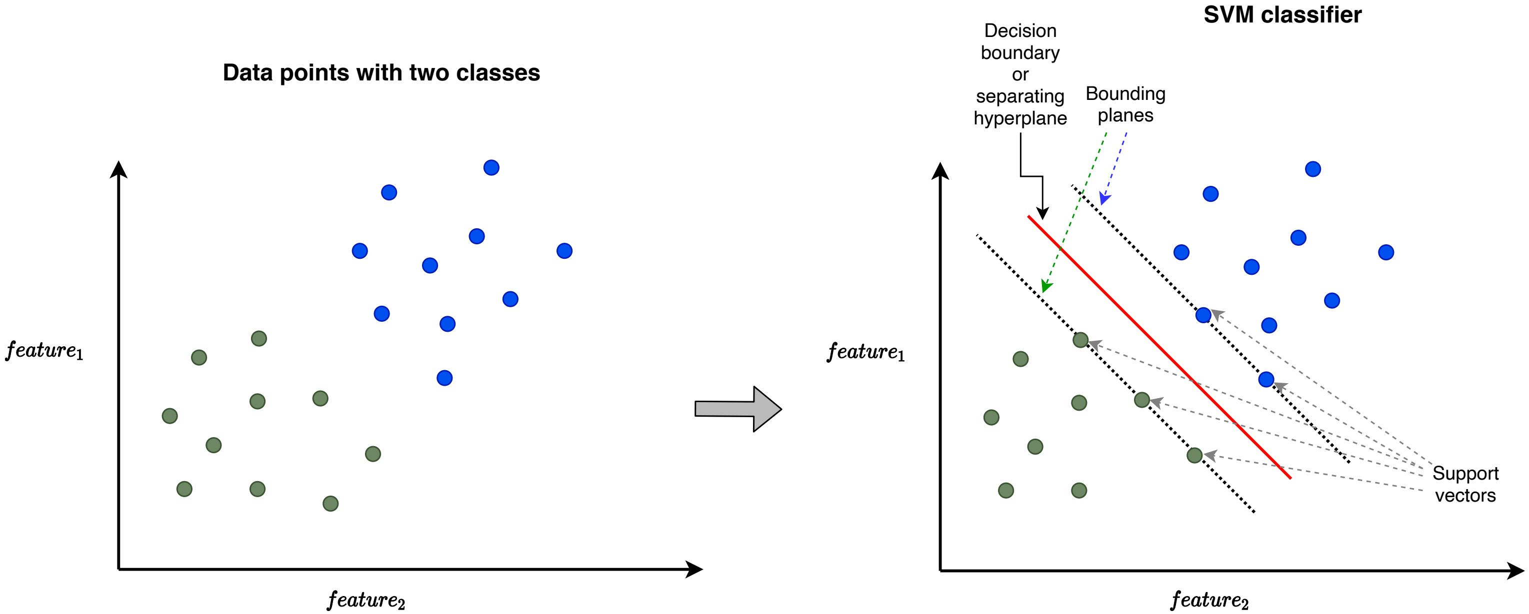
Loss function

Let's learn a few concepts before setting up the loss function.



There are three components here:

- Separating hyperplane
- Bounding planes
- Support vectors



Separating hyperplanes is the **classifier**. It is at equal distance from both the classes and separates them such that there is maximum margin between two classes.

Bounding planes are parallel to separating hyperplanes on the either sides and pass through support vectors.

Support vectors are subset of training points closer to the separating hyperplane and influence its position and orientation

Bounding planes

The bounding planes are defined as follows:

The bounding plane on the side of the positive class has the following equation:

$$\mathbf{w}^T\mathbf{x} + b = 1$$

The bounding plane on the side of the negative class has the following equation:

$$\mathbf{w}^T\mathbf{x} + b = -1$$

We can write this in one equation as follows using the label of an example.

$$y(\mathbf{w}^T\mathbf{x} + b) = 1$$

Any point on or above the bounding plane belongs to the positive class:

$$\mathbf{w}^T\mathbf{x} + b \geq 1$$

Any point on or below the bounding plane belongs to the negative class:

$$\mathbf{w}^T\mathbf{x} + b \leq -1$$

Compactly, the correctly classified points satisfy the following equation:

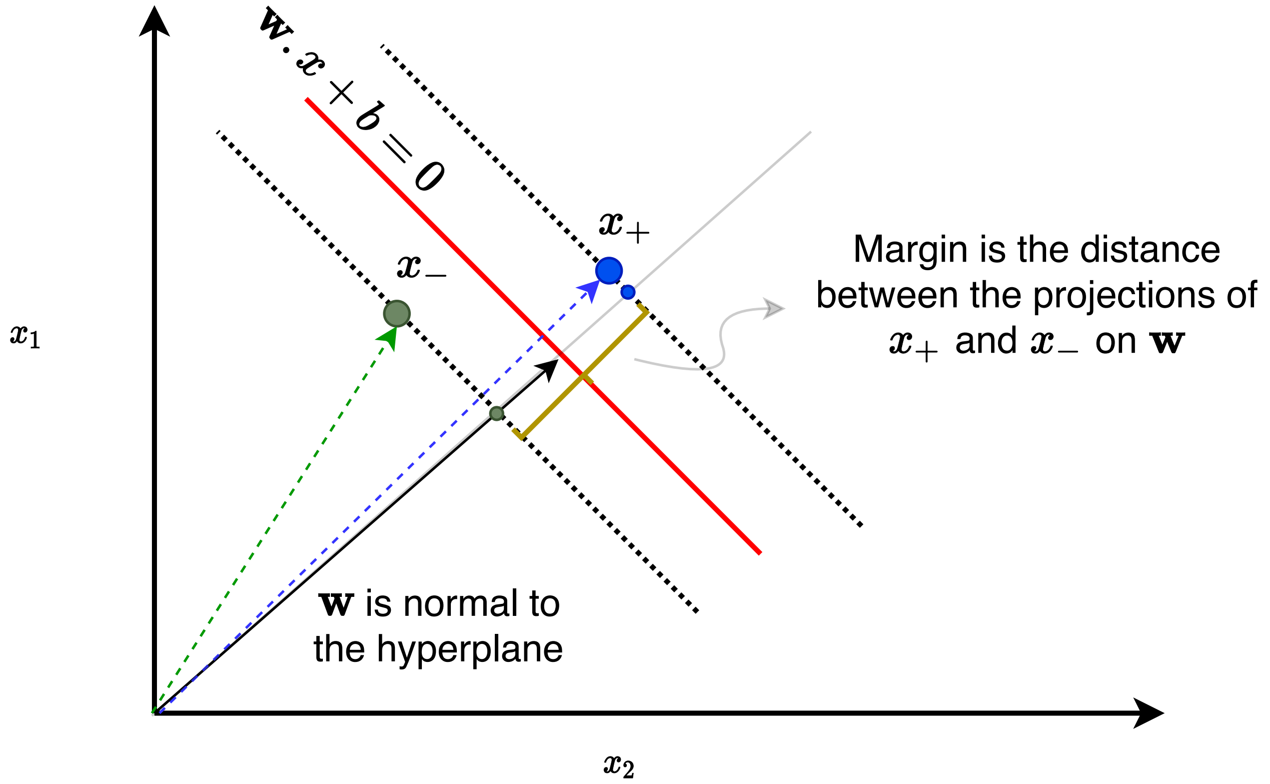
$$y(\mathbf{w}^T\mathbf{x} + b) \geq 1$$

This constraint ensures that none of the points falls within the margin.

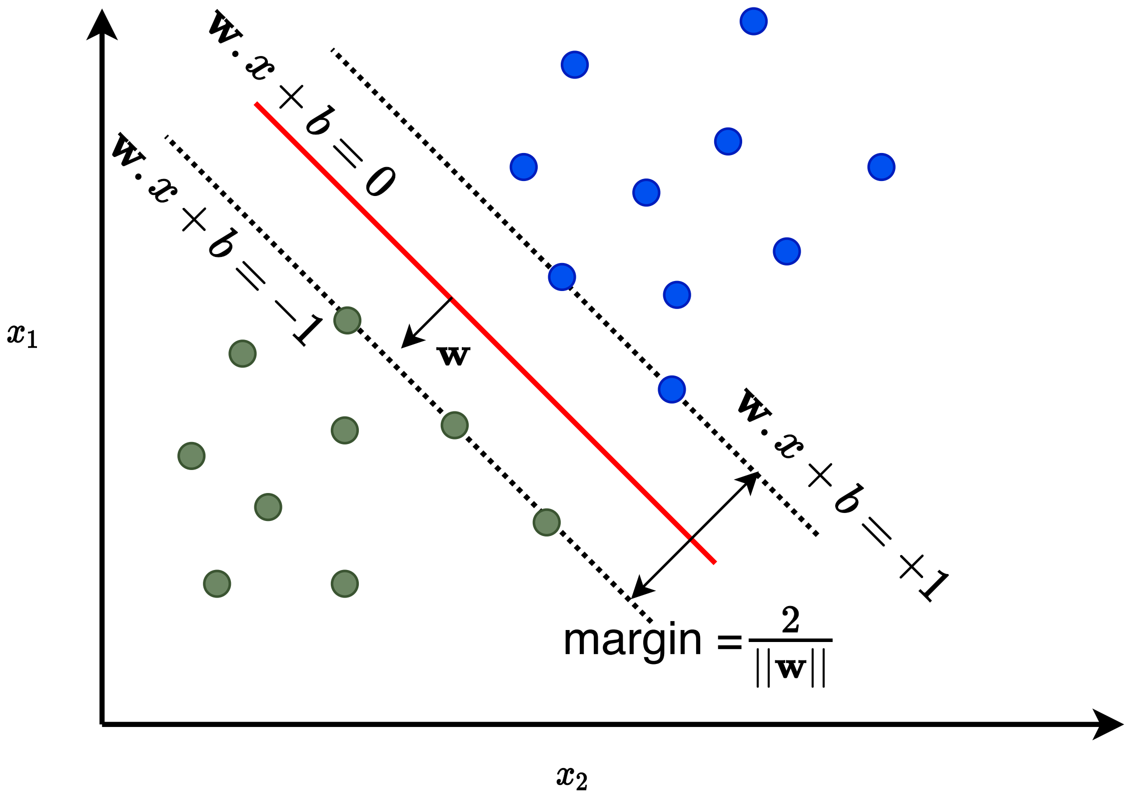
Support vectors

Support vectors are points on the bounding planes.

$$y(\mathbf{w}^T\mathbf{x} + b) = 1$$



Margin



Width of margin is the projection of $(\mathbf{x}_+ - \mathbf{x}_-)$ on unit normal vector $\frac{\mathbf{w}}{||\mathbf{w}||}$. Mathematically,

$$\text{width} = (\mathbf{x}_+ - \mathbf{x}_-) \cdot \frac{\mathbf{w}}{||\mathbf{w}||}$$

For positive support vector $\mathbf{x}_+$: (Using dot product between two vectors here)

$$\mathbf{w}.\mathbf{x}_+ + b = +1$$

$$\mathbf{w}.\mathbf{x}_+ = 1 - b$$

For negative support vector $\mathbf{x}_-$:

$$\mathbf{w}.\mathbf{x}_- + b = -1$$

$$\mathbf{w}.\mathbf{x}_- = -1 - b$$

$$
\begin{aligned}
\text{width} &= (\mathbf{x}_{+} - \mathbf{x}_{-}) . \frac{\mathbf{w}}{\|\mathbf{w}\|} \\
&= \frac{\mathbf{w}.\mathbf{x}_{+} - \mathbf{w}.\mathbf{x}_{-}}{\|\mathbf{w}\|} \\
&= \frac{1 - b - (-1 - b)}{\|\mathbf{w}\|} \\
&= \frac{1 - b + 1 + b}{\|\mathbf{w}\|} \\
&= \frac{2}{\|\mathbf{w}\|}
\end{aligned}
$$

Our objective is to maximize the margin, $\frac{2}{||\mathbf{w}||}$, which is equivalent to minimizing

$$||\mathbf{w}|| = \frac{1}{2}||\mathbf{w}||^2$$

Therefore the optimization problem of linear SVM is written as follows:

$$\min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||^2$$

such that $\quad y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1, i = 1, \ldots, n$

This is called a primal problem and is guaranteed to have a global minimum.

Let's solve this problem with an optimization procedure.

Optimizing SVM Primal Problem

The optimization problem of linear SVM is as follows:

$$\min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||^2$$

such that $$y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1, i = 1, \ldots, n$$

- This is quadratic optimization problem: quadratic objective with linear constraint.
- Can be efficiently solved with QCPC (Quadratically constrained quadratic program) solvers

Dual of SVM primal problem

- Alternatively we can optimize the dual of the primal problem.
- For that we will make use of Lagrange multipliers.

$$J_p(\mathbf{w}, b, \alpha) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{n} \alpha^{(i)} \left(y^{(i)}(\mathbf{w}.\mathbf{x}^{(i)} + b) - 1\right)$$

subject to $\alpha^{(i)} \geq 0 \;\; \forall i = 1, \ldots, n$

This is called Lagrangian function of the SVM, which is differentiable with respect to $\mathbf{w}$ and b.

$$\frac{\partial}{\partial \mathbf{w}} J(\mathbf{w}, b, \alpha) = \mathbf{w} - \sum_{i=1}^{n} \alpha^{(i)} y^{(i)} \mathbf{x}^{(i)} = 0$$ which implies $\mathbf{w} = \sum_{i=1}^{n} \alpha^{(i)} y^{(i)} \mathbf{x}^{(i)}$

$$\frac{\partial}{\partial b} J(\mathbf{w}, b, \alpha) = \sum_{i=1}^{n} \alpha^{(i)} y^{(i)} = 0$$

We have $J(\mathbf{w}, b, \alpha) = \frac{1}{2}||\mathbf{w}||^2 - \sum \alpha^{(i)} \left(y^{(i)}(\mathbf{w}.\mathbf{x}^{(i)} + b) - 1\right)$

$\sum_{i=1}^{n} \alpha^{(i)} y^{(i)} = 0$ $\mathbf{w} = \sum_{i=1}^{n} \alpha^{(i)} y^{(i)} \mathbf{x}^{(i)}$

Substituting these in Lagrangian function of SVM, we get dual problem.

$$
\begin{aligned}
J_d(\alpha) &= \sum_{i=1}^{n} \alpha^{(i)} - \frac{1}{2} \sum_{i=1}^{n} \sum_{k=1}^{n} \alpha^{(i)} \alpha^{(k)} y^{(i)} y^{(k)} x^{(i)^T} x^{(k)} \\
&= \sum_{i=1}^{n} \alpha^{(i)} - \frac{1}{2} \sum_{i=1}^{n} \sum_{k=1}^{n} \langle \alpha^{(i)} y^{(i)} x^{(i)}, \alpha^{(k)} y^{(k)} x^{(k)} \rangle
\end{aligned}
$$

such that $\alpha^{(i)} \geq 0, i \in 1, \ldots, n$

$\sum_{i=1}^{n} \alpha^{(i)} y^{(i)} = 0$

This is a concave problem that is **maximized** using a solver.

$$
\begin{aligned}
J_d(\alpha) &= \sum_{i=1}^{n} \alpha^{(i)} - \frac{1}{2} \sum_{i=1}^{n} \sum_{k=1}^{n} \alpha^{(i)} \alpha^{(k)} y^{(i)} y^{(k)} x^{(i)^T} x^{(k)} \\
&= \sum_{i=1}^{n} \alpha^{(i)} - \frac{1}{2} \sum_{i=1}^{n} \sum_{k=1}^{n} \langle \alpha^{(i)} y^{(i)} x^{(i)}, \alpha^{(k)} y^{(k)} x^{(k)} \rangle
\end{aligned}
$$

such that

$$
\begin{gathered}
\alpha^{(i)} \geq 0, i \in 1, \ldots, n \\
\sum_{i=1}^{n} \alpha^{(i)} y^{(i)} = 0
\end{gathered}
$$

The dual problem is easier to solve as it is expressed in terms of the Lagrange multipliers.

The dual problem depends on the inner product of training data.

Strong duality requires Karush–Kuhn–Tucker (KKT) condition:

$$\alpha^{(i)}\left(\mathbf{y}^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)}+b)-1\right)=0, \forall i \in 1,\ldots,n$$

This implies

If $\alpha^{(i)} > 0$ then $\mathbf{y}^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)}+b)=1$. The data point is a support vector. In other words, it is located on one of the bounding planes.

If $\mathbf{y}^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)}+b)>1$, the distance between data point and the separating hyperplane is more than the margin.

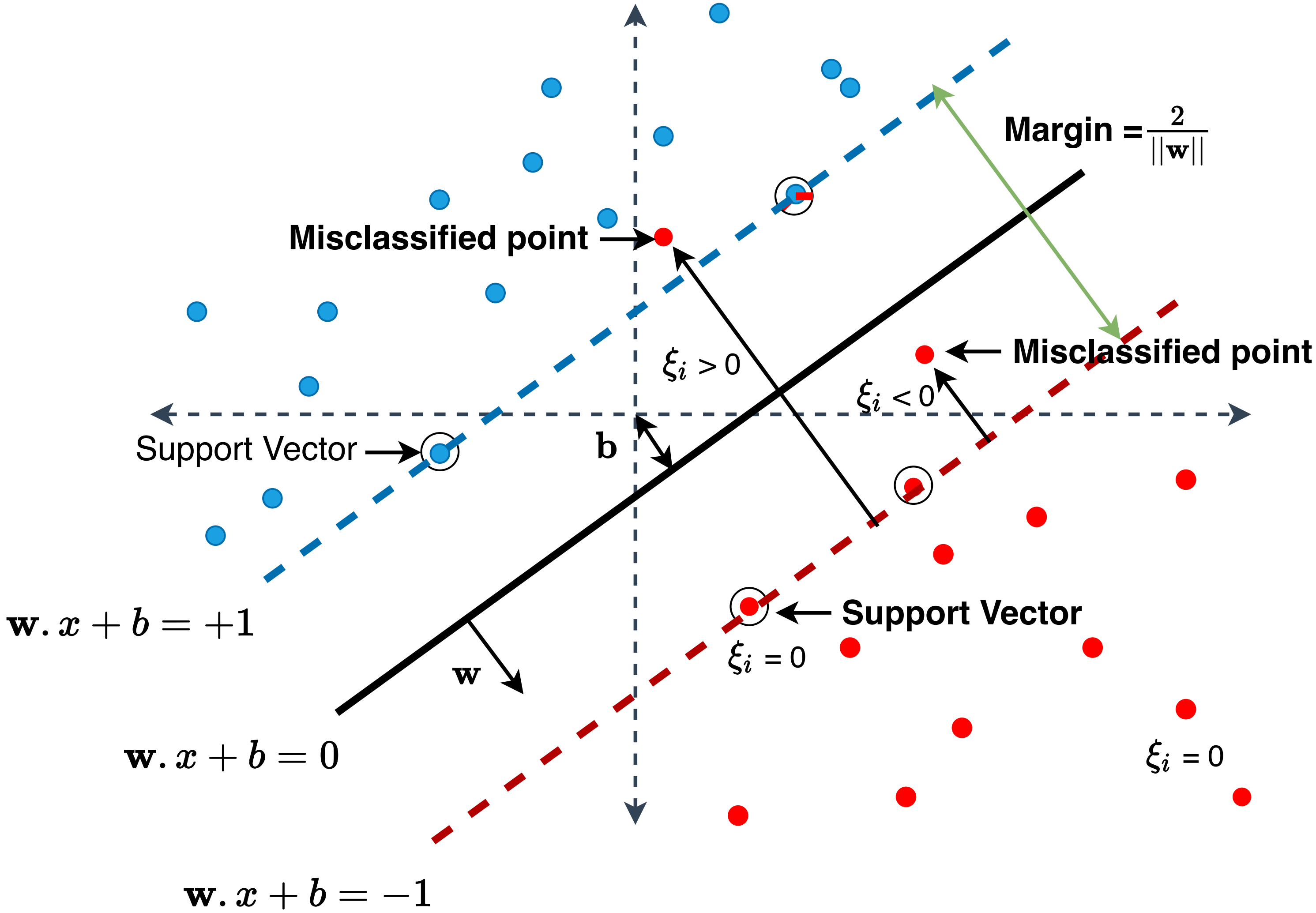
Both primal and dual problems can be solved with optimization solvers to obtain the separating hyperplane for SVMs.

This flavour of SVM where classes are linearly separable is called hard margin SVM

Soft margin SVMs

- The classes are largely linearly separable, but there are a few misclassifications or a few points lie with in margin.
- We are unable to find a perfect hyperplane that maximizes the margin.
- We would like to make some adjustments to the loss function so as to learn a hyperplane with tolerance to a small number of misclassified examples.

Soft margin SVMs



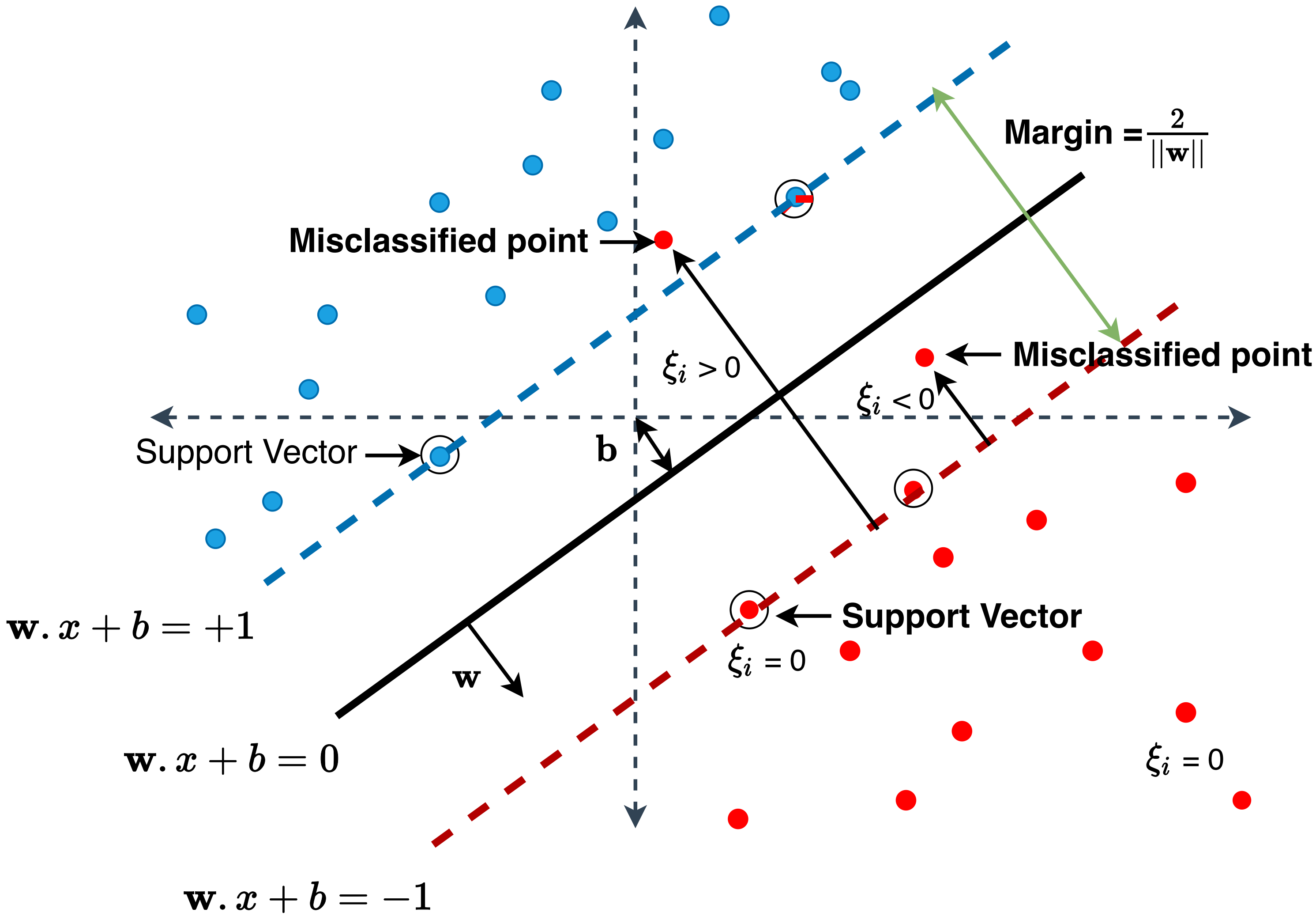
Slack variable

We introduce a slack variable $\xi^{(i)}$ for each training point in the constraint as follows:

$$
\begin{aligned}
y^{(i)}(\mathbf{w}^T \mathbf{x}^{(i)} + b) &\geq 1 - \xi^{(i)}; \\
\xi^{(i)} &> 0
\end{aligned}
$$

The constraints are now a less-strict because each training point $\mathbf{x}^{(i)}$ need only be at a distance of $1 - \xi^{(i)}$ from the separating hyperplane instead of a hard distance of 1.

Soft margin SVMs



In order to prevent slack variable becoming too large, we penalize it in the objective function

$$\min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi^{(i)}$$

such that

$$y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1 - \xi^{(i)}$$
$$\forall_i \xi^{(i)} \geq 0$$

- Slack allows input to be closer to the hyperplane or even be on the wrong side.
- C is large - SVM becomes strict and tries to get all points to the correct side of the hyperplane.
- C is small - SVM slacks and allows many misclassifications or point to lie with in margin.

Let's derive an unconstrained formulation.

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \xi^{(i)}$$

For $C \neq 0$, our objective is to minimize $\xi^{(i)}$ as much as possible, which is possible with $y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) = 1 - \xi^{(i)}$

$$\xi^{(i)} = \begin{cases} 1 - \mathbf{y}^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) & \text{if } \mathbf{y}^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) < 1 \\ 0 & \text{if } \mathbf{y}^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1 \end{cases}$$

We add a non-zero slack $1 - \mathbf{y}^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b)$ for misclassified points or points inside margin.

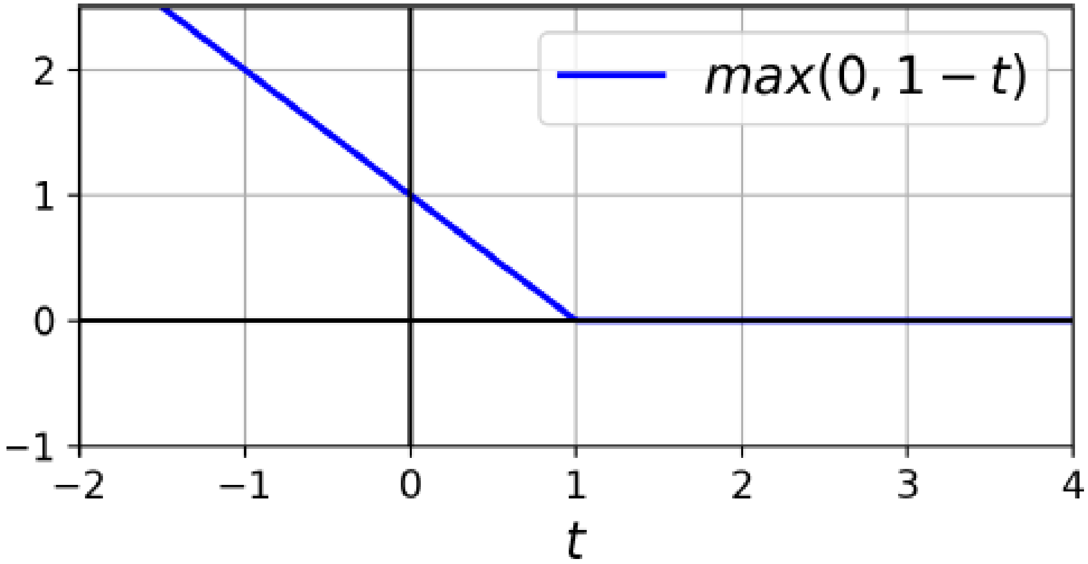
This is equivalent to

$$\xi^{(i)} = \max\left(1 - \mathbf{y}^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b), 0\right)$$

Let's plug this in the soft margin SVM objective function.

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \max\left(1 - \mathbf{y}^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b), 0\right)$$

- The second term is the hinge loss.



x-axis is the $t = y^{(i)}(\mathbf{w}.\mathbf{x}^{(i)} + b)$ and y-axis is the misclassification cost.

We need to minimize the soft margin loss to find the max-margin classifier.

$$\frac{\partial}{\partial \mathbf{w}} J(\mathbf{w}, b) = \frac{\partial}{\partial \mathbf{w}} \left(\frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{n} \max\left(1 - \mathbf{y}^{(i)} (\mathbf{w}^T \mathbf{x}^{(i)} + b), 0\right) \right)$$

The partial derivative of the second term depends on the misclassification penalty:

$$\frac{\partial}{\partial \mathbf{w}} \max\left(0, [1 - \mathbf{y}^{(i)} (\mathbf{w}^T \mathbf{x}^{(i)} + b)]\right) = \begin{cases} \mathbf{0} & \text{if } \max\left(0, [1 - \mathbf{y}^{(i)} (\mathbf{w}^T \mathbf{x}^{(i)} + b)]\right) = 0 \\ \mathbf{y}^{(i)} \mathbf{x}^{(i)} & \text{otherwise} \end{cases}$$

$$\frac{\partial}{\partial b} \max\left(0, [1 - \mathbf{y}^{(i)} (\mathbf{w}^T \mathbf{x}^{(i)} + b)]\right) = \begin{cases} \mathbf{0} & \text{if } \max\left(0, [1 - \mathbf{y}^{(i)} (\mathbf{w}^T \mathbf{x}^{(i)} + b)]\right) = 0 \\ \mathbf{y}^{(i)} & \text{otherwise} \end{cases}$$

Partial derivatives

$$\frac{\partial}{\partial \mathbf{w}} \max\left(0, [1 - \mathbf{y}^{(i)}(\mathbf{w}^T \mathbf{x}^{(i)} + b)]\right) = \begin{cases} \mathbf{0} & \text{if } \max\left(0, [1 - \mathbf{y}^{(i)}(\mathbf{w}^T \mathbf{x}^{(i)} + b)]\right) = 0 \\ \mathbf{y}^{(i)}\mathbf{x}^{(i)} & \text{otherwise} \end{cases}$$

$$\frac{\partial}{\partial b} \max\left(0, [1 - \mathbf{y}^{(i)}(\mathbf{w}^T \mathbf{x}^{(i)} + b)]\right) = \begin{cases} \mathbf{0} & \text{if } \max\left(0, [1 - \mathbf{y}^{(i)}(\mathbf{w}^T \mathbf{x}^{(i)} + b)]\right) = 0 \\ \mathbf{y}^{(i)} & \text{otherwise} \end{cases}$$

Writing compactly:

$$\frac{\partial}{\partial \mathbf{w}} J(\mathbf{w}, b) = \mathbf{w} + C \sum_{i=1}^{n} \mathbf{1}\left(1 - \mathbf{y}^{(i)}(\mathbf{w}^T \mathbf{x}^{(i)} + b) > 0\right) \mathbf{y}^{(i)} \mathbf{x}^{(i)}$$

$$\frac{\partial}{\partial b} J(\mathbf{w}, b) = C \sum_{i=1}^{n} \mathbf{1}\left(1 - \mathbf{y}^{(i)}(\mathbf{w}^T \mathbf{x}^{(i)} + b) > 0\right) \mathbf{y}^{(i)}$$

Gradient descent update rule

$$\mathbf{w}^{(\text{new})} = \mathbf{w}^{(\text{old})} - \text{learning rate } \times \left(\mathbf{w} + C \sum_{i=0}^{n} \mathbf{1}\left(1 - \mathbf{y}^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) > 0\right)\mathbf{y}^{(i)}\mathbf{x}^{(i)}\right)$$

$$b^{(\text{new})} = b^{(\text{old})} - \text{learning rate } \times C \sum_{i=0}^{n} \mathbf{1}\left(1 - \mathbf{y}^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) > 0\right)\mathbf{y}^{(i)}$$

Evaluation measures

- Usual classification evaluation measures like precision, recall, f1-score and accuracy.

Kernel SVMs

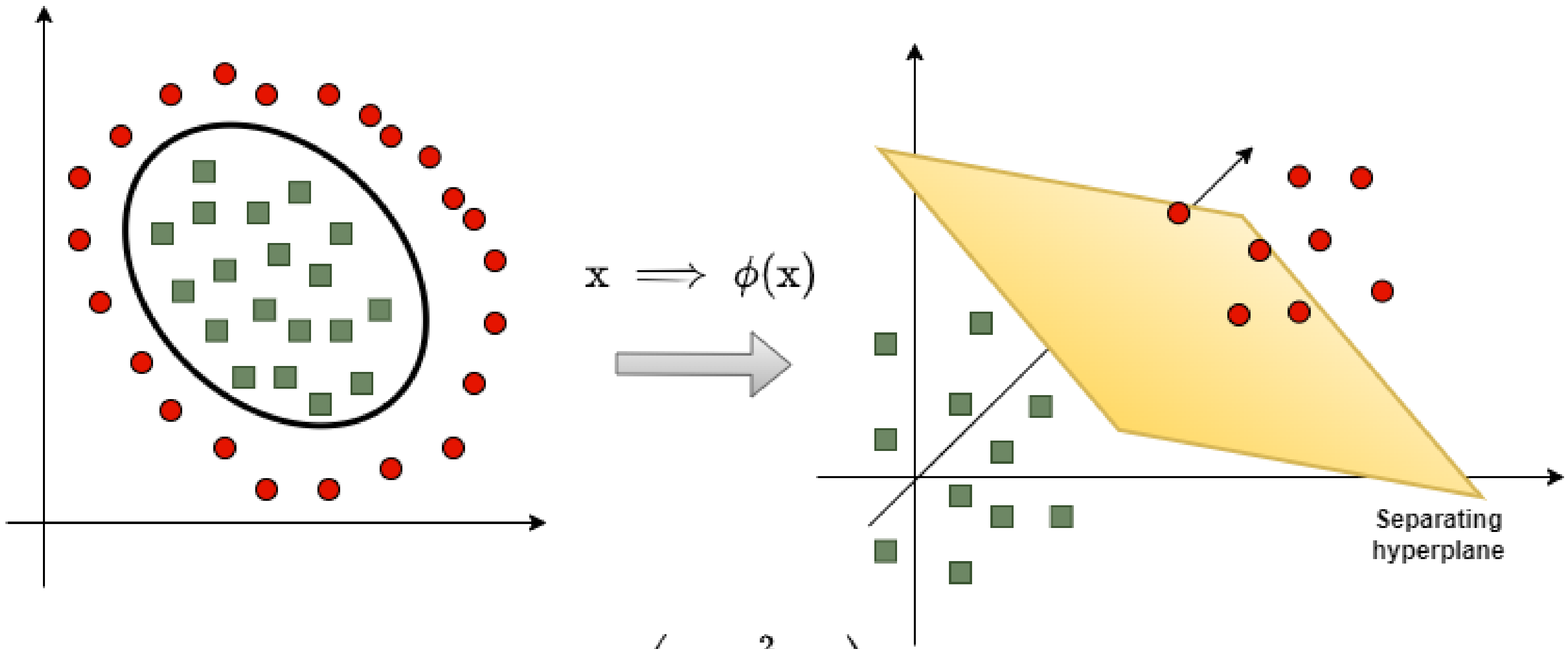
Basic Idea

- Kernel SVM is used for non-linearly separable data.

- Remember that so far we were performing non-linear transformation on the input feature space (e.g. polynomial transformation) and then training the model in the transformed space for learning non-linear decision boundaries.

- Kernel SVM computes dot product in transformed feature space, but **without explicitly calculating the transformation**.

From low dimension to higher dimension



$$\phi(\mathrm{x}) = \phi \begin{pmatrix} x_1^2 \\ \sqrt{2}\, x_1 x_2 \\ x_2^2 \end{pmatrix}$$

Recall SVM dual objective function

$$J_d(\alpha) = \sum_{i=1}^{n} \alpha^{(i)} - \frac{1}{2} \sum_{i=1}^{n} \sum_{k=1}^{n} \alpha^{(i)} \alpha^{(k)} \mathbf{y}^{(i)} \mathbf{y}^{(k)} \mathbf{x}^{(i)T} \mathbf{x}^{(k)}$$

Writing this with kernel function

$$= \sum_{i=1}^{n} \alpha^{(i)} - \frac{1}{2} \sum_{i=1}^{n} \sum_{k=1}^{n} \alpha^{(i)} \alpha^{(k)} \mathbf{y}^{(i)} \mathbf{y}^{(k)} K(\mathbf{x}^{(i)}, \mathbf{x}^{(k)})$$

such that

$$\alpha^{(i)} \geq 0, i \in 1, \ldots, n$$
$$\sum_{i=1}^{n} \alpha^{(i)} \mathbf{y}^{(i)} = 0$$

Kernel performs dot product between input feature vectors in high dimensional space without actually projecting or transforming the input features in that space.

Linear kernel:

$$K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \mathbf{x}^{(i)}{}^{T}\mathbf{x}^{(j)}$$

Polynomial kernel:

$$K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \left(1 + \mathbf{x}^{(i)}{}^{T}\mathbf{x}^{(j)}\right)^{d}$$

Radial basis functions (RBF)

$$K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \exp\left(-\frac{(\mathbf{x}^{(i)} - \mathbf{x}^{(j)})^{2}}{\sigma^{2}}\right)$$

Demonstrating kernel trick with polynomial

$$
\begin{aligned}
\phi(\mathbf{a})^T \phi(\mathbf{b}) &= \begin{pmatrix} a_1^2 \\ \sqrt{2}\, a_1 a_2 \\ a_2^2 \end{pmatrix}^T \begin{pmatrix} b_1^2 \\ \sqrt{2}\, b_1 b_2 \\ b_2^2 \end{pmatrix} \\
&= a_1^2 b_1^2 + 2 a_1 b_1 a_2 b_2 + a_2^2 + b_2^2 \\
&= (a_1 b_1 + a_2 b_2)^2 \\
&= \left(\begin{pmatrix} a_1 \\ a_2 \end{pmatrix}^T \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \right)^2 \\
&= (\mathbf{a}^T \mathbf{b})^2
\end{aligned}
$$

Model with kernel SVM

$$h(\mathbf{x}) = \operatorname{sign}\left(\sum_{i=1}^{n} \alpha^{(i)} \mathbf{y}^{(i)} K(\mathbf{x}^{(i)}, \mathbf{x}) + b\right)$$