AI and Machine Learning

Zhiyun Lin

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Support Vector Machine

- Margin
- Max-margin Classification
- Kernel Tricks

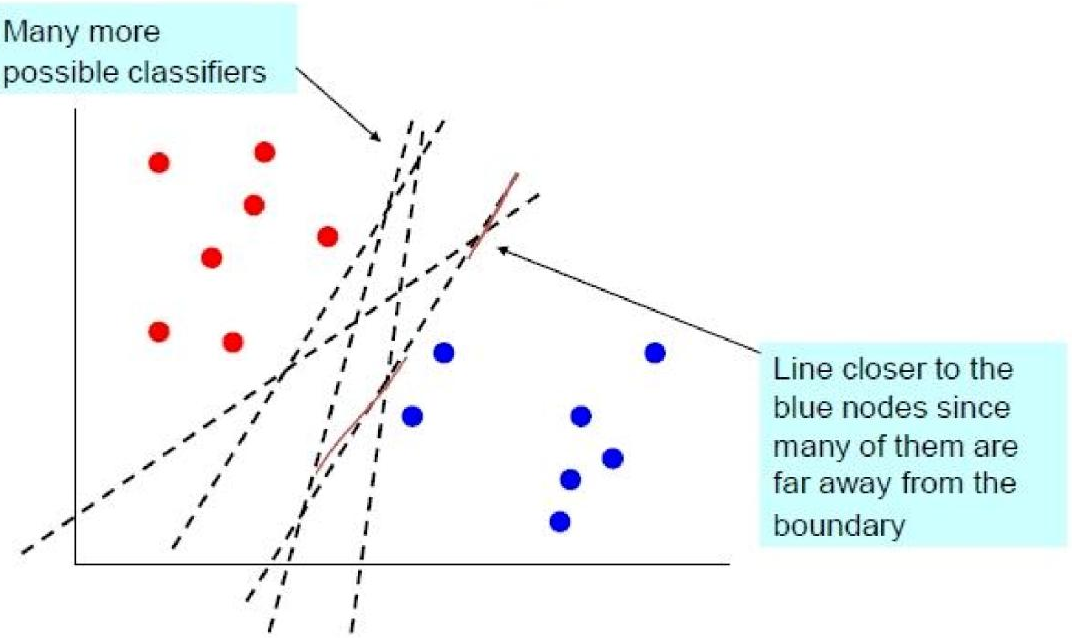
Support Vector Machine

- We are back to **supervised learning**
- We are given training data $\{x^{(i)}, t^{(i)}\}_{i=1}^{N}$
- We will look at **classification**, so $t^{(i)}$ will represent the class label
- We will focus on **binary classification** (two classes)
- We will consider a **linear classifier** first

> Tiny change frombefore: instead of using $t = 1$ and $t = 0$ for positive and negative class, we will use $t = 1$ for the positive and $t = -1$ for the negative class

Logistic Regression

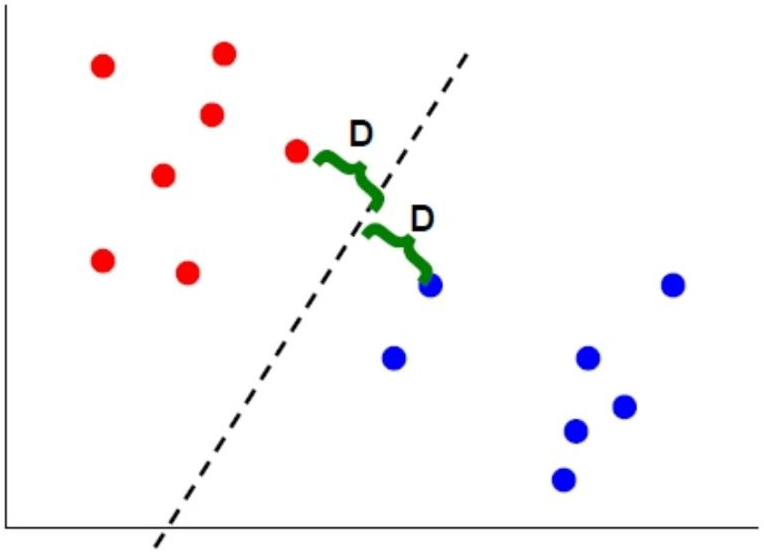
- Recall logistic regression classifiers



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

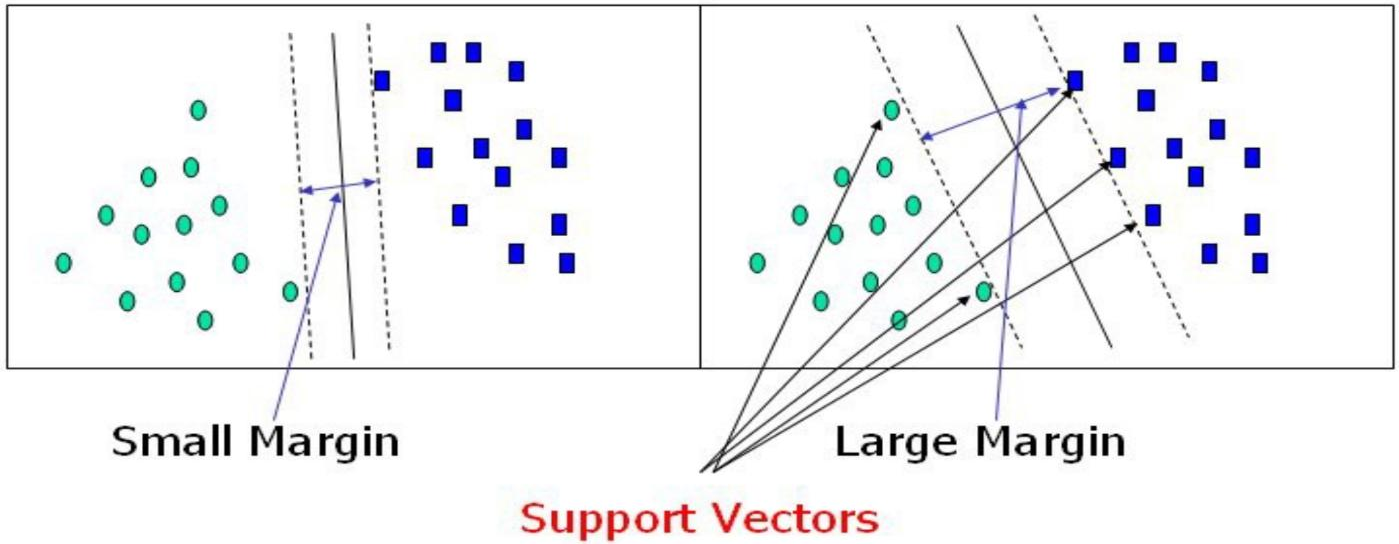
Max Margin Classification

- Instead of fitting all the points, focus on the boundary points
- **Aim:** Learn a boundary that leads to the largest **margin** (buffer) from points on both sides



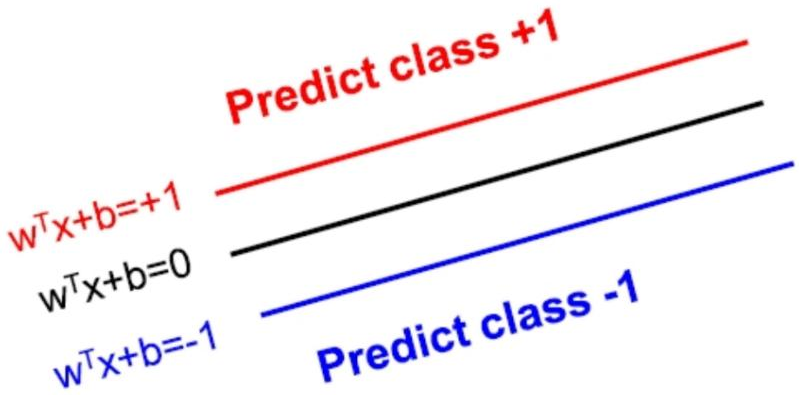
Max Margin Classification

- **Why:** intuition; theoretical support; and works well in practice
- Subset of vectors that support (determine boundary) are called the **support vectors**



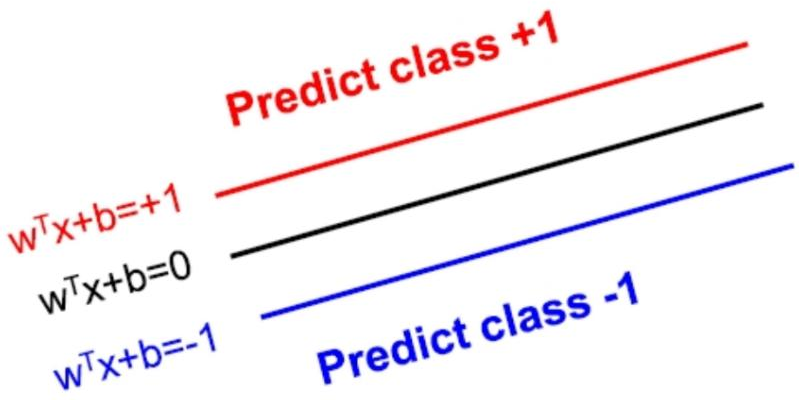
Linear SVM

- **Max margin classifier:** Inputs in margin are of unknown class



$$y = \begin{cases} 1 & \text{if } \mathbf{w}^T\mathbf{x} + b \geq 1 \\ -1 & \text{if } \mathbf{w}^T\mathbf{x} + b \leq -1 \\ \text{Undefined} & \text{if } -1 \leq \mathbf{w}^T\mathbf{x} + b \leq 1 \end{cases}$$

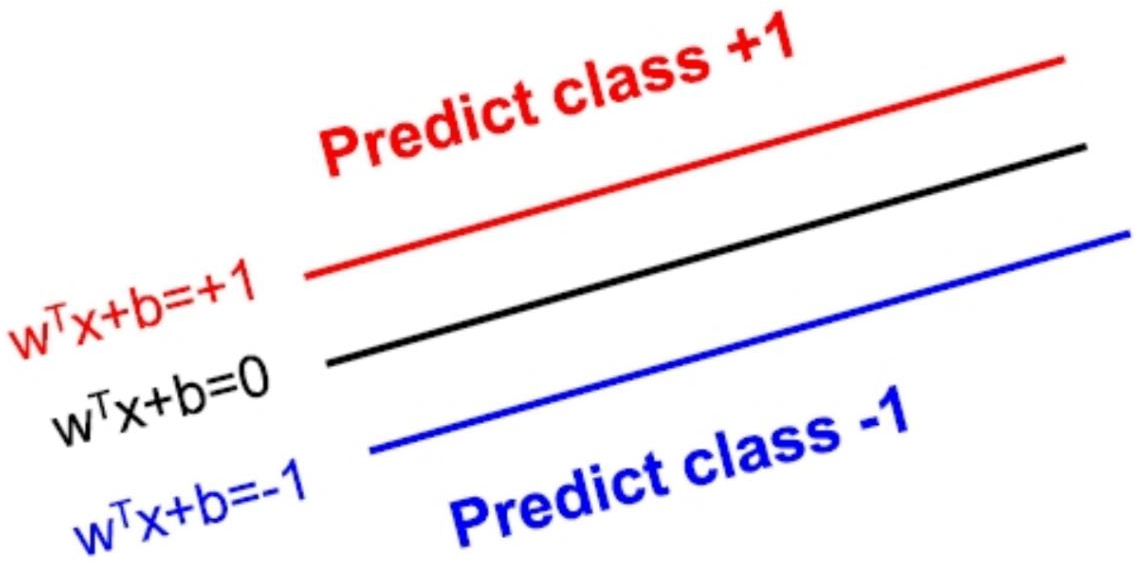
Linear SVM



$$y = \begin{cases} 1 & \text{if } \mathbf{w}^T\mathbf{x} + b \geq 1 \\ -1 & \text{if } \mathbf{w}^T\mathbf{x} + b \leq -1 \\ \text{Undefined} & \text{if } -1 \leq \mathbf{w}^T\mathbf{x} + b \leq 1 \end{cases}$$

> Equivalently, can write above condition as:
>
> $$(w^T x + b)y \geq 1$$

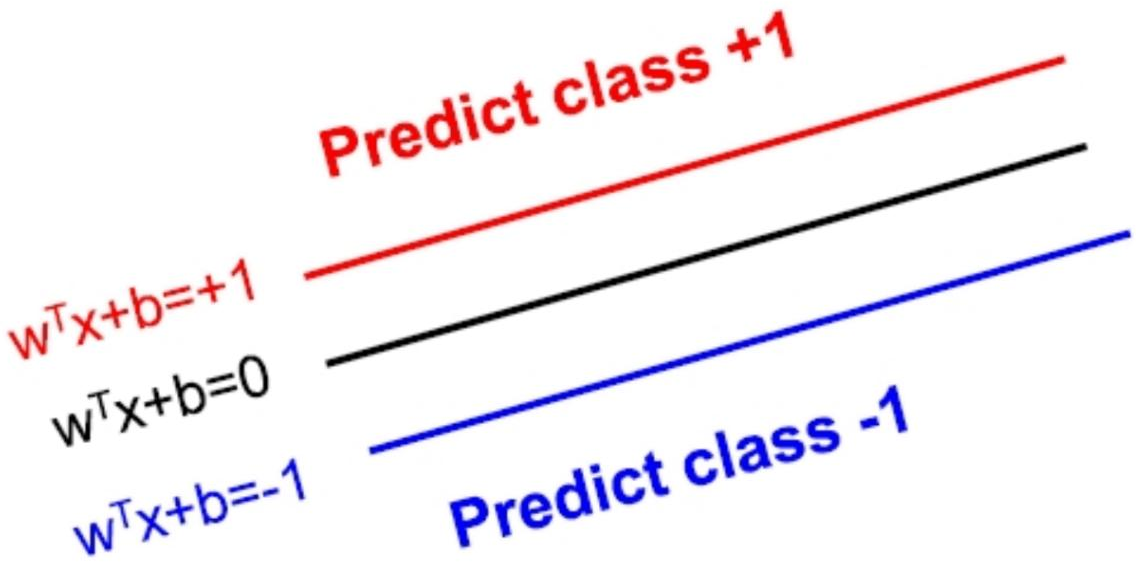
Geometry of the Problem



- The vector w is orthogonal to the +1 plane.

> If u and v are two points on that plane, then $w^T(u-v)=0$

Geometry of the Problem



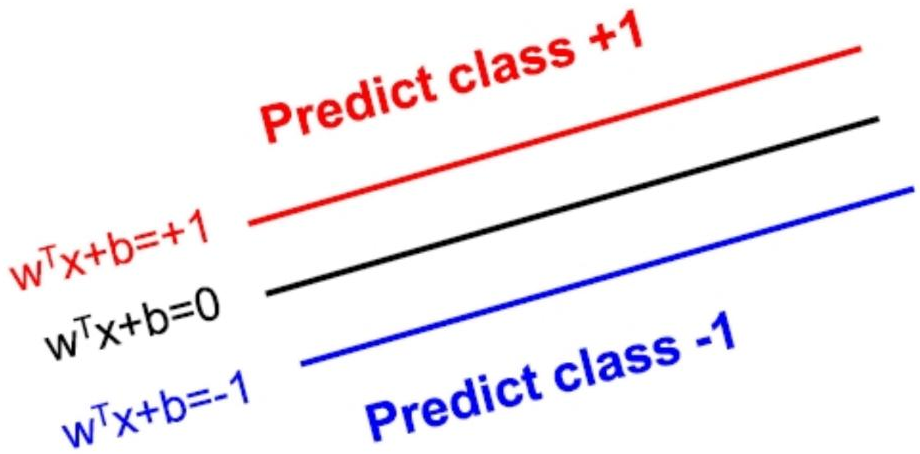
- Same is true for -1 plane
- Also: for point x_+ on +1 plane and x_- nearest point on -1 plane:

$$x_+ = \lambda w + x_-$$

Computing the Margin

- Also: for point x_+ on +1 plane and x_- nearest point on -1 plane:

$$x_+ = \lambda w + x_-$$



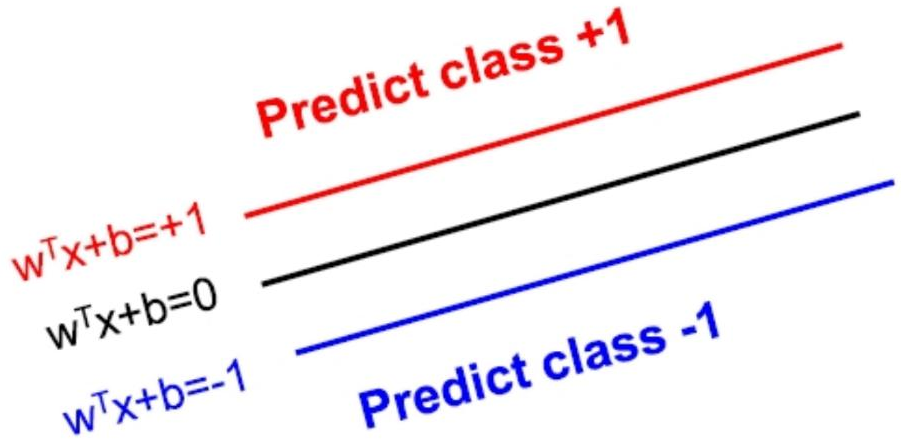
$$w^T x_+ + b = 1 \quad \Rightarrow \quad w^T(\lambda w + x_-) + b = 1$$
$$\Rightarrow \quad w^T x_- + b + \lambda w^T w = 1 \quad \Rightarrow \quad -1 + \lambda w^T w = 1$$

- **Therefore**,

$$\lambda = \frac{2}{w^T w}$$

Computing the Margin

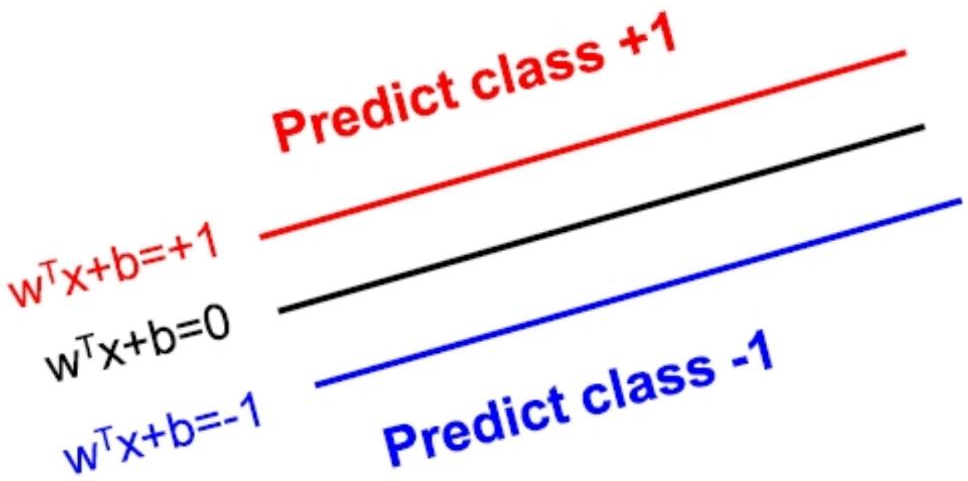
- Define the margin M to be the distance between the +1 and -1 planes
- We can now express this in terms of w to **maximize the margin**
  - Or, equivalently, we **minimize the length** of w



> $$M = \|x_+ - x_-\| = \|\lambda w\| = \lambda\sqrt{w^T w} = 2\frac{\sqrt{w^T w}}{w^T w} = \frac{2}{\|w\|}$$

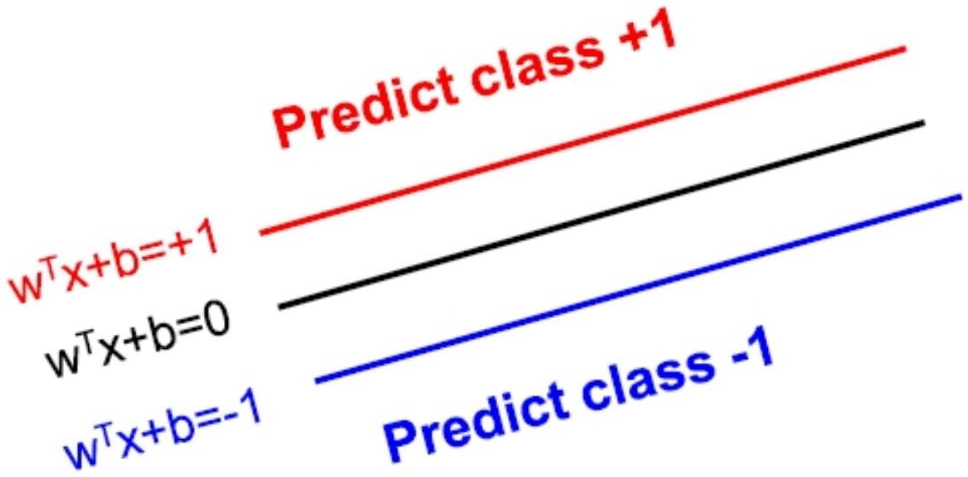
Learning a Margin-Based Classifier

- We can search for the optimal w and b by finding a solution that:
  i. Correctly classifies the training examples: $\{(x^{(i)}, t^{(i)})\}_{i=1}^{N}$
  ii. Maximize the margin (same as minimizing $w^T w$)



> $$\min_{w,b} \frac{1}{2}\|w\|^2$$
> $$\text{s.t.} \quad (w^T x^{(i)} + b)t^{(i)} \geq 1, \quad \forall i = 1, \ldots, N$$

Learning a Margin-Based Classifier



> $$\min_{w,b} \frac{1}{2}\|w\|^2$$
> $$\text{s.t.} \quad (w^T x^{(i)} + b)t^{(i)} \geq 1, \quad \forall i = 1, \ldots, N$$

- This is called the **primal formulation** of Support Vector Machine (SVM)
- Apply **Lagrange multipliers**: formulate equivalent problem

Detour: Lagrange Multipliers

Constrained Optimization

- Constrained optimization with **equality constraints**

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{s.t.} \quad & h_i(x) = 0, \ \ \forall i = 1, \dots, m \end{aligned}$$

- Convert to an unconstrained optimization problem:

$$\min_{x} \max_{\mu_i \neq 0} l(x, \mu)$$

> where $l(x, \mu) = f(x) + \sum_{i=1}^{m} \mu_i h_i(x)$ is the **Lagrangian** function, and $\mu_i, \ i = 1, \dots, m$ are the **Lagrange multipliers**

Intuition Behind

- Constrained optimization with **equality constraints**

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{s.t.} \quad & h_i(x) = 0, \ \ \forall i = 1, \dots, m \end{aligned}$$

- To remove the equality constraint, we add a **penalty term** $p_i(x)$:

$$\min_{x} \quad f(x) + \sum_{i=1}^{m} p_i(x)$$

- What is the appropriate penalty term?

$$p_i(x) = \begin{cases} 0 & \text{if} \quad h_i(x) = 0 \\ \infty & \text{otherwise} \end{cases}$$

- A good choice is:

$$p_i(x) = \max_{\mu_i \neq 0} \mu_i h_i(x)$$

Intuition Behind

- Therefore, it is equivalently to have:

$$\begin{aligned}
\min_x \quad \left\{ f(x) + \sum_{i=1}^{m} p_i(x) \right\} &= \min_x \quad \left\{ f(x) + \sum_{i=1}^{m} \max_{\mu_i \neq 0} \mu_i h_i(x) \right\} \\
&= \min_x \max_{\mu_i \neq 0} \quad \left\{ f(x) + \sum_{i=1}^{m} \mu_i h_i(x) \right\} \\
&= \min_x \max_{\mu_i \neq 0} \quad l(x, \mu)
\end{aligned}$$

Constrained Optimization

- Constrained optimization with **inequality constraints**

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{s.t.} \quad & g_i(x) \leq 0, \ \ \forall i = 1, \ldots, n \end{aligned}$$

- Convert to an unconstrained optimization problem:

$$\min_{x} \max_{\alpha_i \geq 0} l(x, \alpha)$$

> where $l(x, \alpha) = f(x) + \sum_{i=1}^{m} \alpha_i g_i(x)$ is the **Lagrangian** function, and $\alpha_i, \ i = 1, \ldots, n$, are the **Lagrange multipliers**

Intuition Behind

- Constrained optimization with **inequality constraints**

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{s.t.} \quad & g_i(x) \leq 0, \ \ \forall i = 1, \ldots, n \end{aligned}$$

- To remove the inequality constraint, we add a **penalty term** $p_i(x)$:

$$\min_{x} \quad f(x) + \sum_{i=1}^{m} p_i(x)$$

- What is the appropriate penalty term?

$$p_i(x) = \begin{cases} 0 & \text{if} \quad g_i(x) \leq 0 \\ \infty & \text{otherwise} \end{cases}$$

- A good choice is:

$$p_i(x) = \max_{\alpha_i \geq 0} \alpha_i g_i(x)$$

Intuition Behind

- Therefore, it is equivalently to have:

$$\begin{aligned}
\min_x \quad \left\{ f(x) + \sum_{i=1}^{n} p_i(x) \right\} &= \min_x \quad \left\{ f(x) + \sum_{i=1}^{n} \max_{\alpha_i \geq 0} \alpha_i g_i(x) \right\} \\
&= \min_x \max_{\alpha_i \geq 0} \quad \left\{ f(x) + \sum_{i=1}^{n} \alpha_i g_i(x) \right\} \\
&= \min_x \max_{\alpha_i \geq 0} \quad l(x, \alpha)
\end{aligned}$$

- This is called the **primal formulation**

Lagrangian Duality

- Primal formulation

$$\min_{x} \max_{\alpha_i \geq 0} \quad l(x, \alpha)$$

- Dual formulation

$$\max_{\alpha_i \geq 0} \min_{x} \quad l(x, \alpha)$$

> It is always true that
>
> $$\max_{\alpha_i \geq 0} \min_{x} \quad l(x, \alpha) \leq \min_{x} \max_{\alpha_i \geq 0} \quad l(x, \alpha)$$

- The equality holds in certain conditions: **KKT Condition**

Max-Min Inequality

> It is always true that
>
> $$\max_y \min_x \quad f(x,y) \leq \min_x \max_y \quad f(x,y)$$

- Let's see why:

$$\min_x f(x,y) \leq f(x,y) \leq \max_y f(x,y), \quad \forall x, y$$

- This means that **for any x and y**,

$$\min_x f(x,y) \leq \max_y f(x,y)$$

- Thus, the maximum of the left is less than or equal to the minimum of the right, i.e.,

$$\max_y \min_x f(x,y) \leq \min_x \max_y f(x,y)$$

Back to Linear SVM

Learning a Linear SVM

- Convert the constrained minimization to an unconstrained optimization problem:
  - represent constraints as penalty terms:

$$\min_{w,b} \frac{1}{2}||w||^2 + \text{penalty-term}$$

- For data $\{(x^{(i)}, t^{(i)})\}_{i=1}^{N}$, use the following penalty

$$\max_{\alpha_i \geq 0} \quad \alpha_i[1 - (w^T x^{(i)} + b)t^{(i)}] = \begin{cases} 0 & \text{if} \quad (w^T x^{(i)} + b)t^{(i)} \geq 1 \\ \infty & \text{otherwise} \end{cases}$$

- Rewrite the minimization problem.

> “
>
> $$\min_{w,b} \left\{ \frac{1}{2}\|w\|^2 + \sum_{i=1}^{N} \max_{\alpha_i \geq 0} \alpha_i \left[1 - (w^T x^{(i)} + b)t^{(i)}\right] \right\}$$

Learning a Linear SVM

- Rewrite the minimization problem.

> “
>
> $$\min_{w,b} \left\{ \frac{1}{2}\|w\|^2 + \sum_{i=1}^{N} \max_{\alpha_i \geq 0} \alpha_i \left[1 - (w^T x^{(i)} + b) t^{(i)} \right] \right\}$$

- Here $\alpha_i,\ i = 1, \ldots, N$, are the **Lagrange multipliers**
- It is equivalent to:

$$\min_{w,b} \max_{\alpha_i \geq 0} \left\{ \frac{1}{2}\|w\|^2 + \sum_{i=1}^{N} \alpha_i \left[1 - (w^T x^{(i)} + b) t^{(i)} \right] \right\}$$

Solution to Linear SVM

- Let:

$$J(w, b; \alpha) = \frac{1}{2}||w||^2 + \sum_{i=1}^{N} \alpha_i \left[1 - (w^T x^{(i)} + b)t^{(i)}\right]$$

- Swap the "max" and "min": This is a lower bound

$$\max_{\alpha_i \geq 0} \min_{w,b} J(w, b; \alpha) \leq \min_{w,b} \max_{\alpha_i \geq 0} J(w, b; \alpha)$$

- Equality holds in certain conditions: **KKT Condition**

Solution to Linear SVM

- Solving

$$\max_{\alpha_i \geq 0} \min_{w,b} J(w,b;\alpha) = \max_{\alpha_i \geq 0} \min_{w,b} \left\{ \frac{1}{2}||w||^2 + \sum_{i=1}^{N} \alpha_i \left[1 - (w^T x^{(i)} + b) t^{(i)} \right] \right\}$$

- First minimize $J()$ w.r.t. $w,\ b$ for fixed Lagrange multipliers:

$$\frac{\partial J(w,b;\alpha)}{\partial w} = w - \sum_{i=1}^{N} \alpha_i x^{(i)} t^{(i)} = 0$$

$$\frac{\partial J(w,b;\alpha)}{\partial b} = -\sum_{i=1}^{N} \alpha_i t^{(i)} = 0$$

- We obtain

$$w = \sum_{i=1}^{N} \alpha_i t^{(i)} x^{(i)}$$

Solution to Linear SVM

> “
>
> $$w = \sum_{i=1}^{N} \alpha_i t^{(i)} x^{(i)}$$

- Then substitute back to get final optimization:

$$\begin{aligned} L &= \max_{\alpha_i \geq 0} \left\{ \sum_{i=1}^{N} \alpha_i + \tfrac{1}{2} \left[\sum_{i=1}^{N} \alpha_i t^{(i)} x^{(i)} \right]^T \left[\sum_{i=1}^{N} \alpha_i t^{(i)} x^{(i)} \right] - \sum_{j=1}^{N} \alpha_j t^{(j)} \left[\sum_{i=1}^{N} \alpha_i t^{(i)} x^{(i)} \right]^T x^{(j)} \right\} \\ &= \max_{\alpha_i \geq 0} \left\{ \sum_{i=1}^{N} \alpha_i - \tfrac{1}{2} \sum_{i,j=1}^{N} t^{(i)} t^{(j)} \alpha_i \alpha_j ({x^{(i)}}^T x^{(j)}) \right\} \end{aligned}$$

Summary of Linear SVM

- Binary and linear separable classification
- Linear classifier with maximal margin
- Training SVM by maximizing

$$\max_{\alpha_i \geq 0} \left\{ \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} t^{(i)} t^{(j)} \alpha_i \alpha_j (x^{(i)^T} x^{(j)}) \right\}$$

$$\text{subject to} \quad \alpha_i \geq 0; \quad \sum_{i=1}^{N} \alpha_i t^{(i)} = 0$$

- The weights are

$$w = \sum_{i=1}^{N} \alpha_i t^{(i)} x^{(i)}$$

Summary of Linear SVM

- The weights are

$$w = \sum_{i=1}^{N} \alpha_i t^{(i)} x^{(i)}$$

- Only a small subset of α_i's will be nonzero (**why?**), and the corresponding $x^{(i)}$'s are the **support vectors** $\mathbf{S}$
- Prediction on a new example:.

$$y = \text{sign}\left[b + x \cdot \left(\sum_{i=1}^{N} \alpha_i t^{(i)} x^{(i)}\right)\right] = \text{sign}\left[b + x \cdot \left(\sum_{i \in \mathbf{S}} \alpha_i t^{(i)} x^{(i)}\right)\right]$$

Optimal Value for b

- Convert the constrained minimization to an unconstrained optimization problem:
  - represent constraints as penalty terms:

$$\min_{w,b} \frac{1}{2}||w||^2 + \text{penalty-term}$$

- For data $\{(x^{(i)}, t^{(i)})\}_{i=1}^{N}$, use the following penalty

$$\max_{\alpha_i \geq 0} \quad \alpha_i[1 - (w^T x^{(i)} + b)t^{(i)}] = \begin{cases} 0 & \text{if} \quad (w^T x^{(i)} + b)t^{(i)} \geq 1 \\ \infty & \text{otherwise} \end{cases}$$

- Let α_i^*, w^*, b^* be the optimal solution to the problem. Then:

> “
>
> $$\alpha_i^* \left[1 - (w^{*T} x^{(i)} + b^*)t^{(i)}\right] = 0$$

Optimal Value for b

> $$\alpha_i^* \left[1 - (w^{*T} x^{(i)} + b^*) t^{(i)} \right] = 0$$

- For nonzero α_i^*, we must have:

$$1 - (w^{*T} x^{(i)} + b^*) t^{(i)} = 0$$

- From above, we can get:

> $$b^* = t^{(i)} - w^{*T} x^{(i)} = t^{(i)} - \sum_{j=1}^{N} \alpha_j^* t^{(j)} \left(x^{(j)} \cdot x^{(i)} \right)$$

> where i is the **index of a support vector**.

Optimal Value for b

- Averaging over all support vectors:

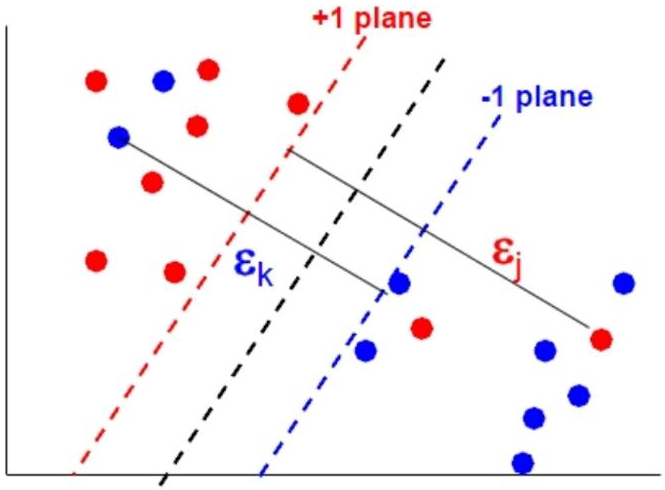
$$b^* = \frac{1}{|\mathbf{S}|} \sum_{i \in \mathbf{S}} \left[t^{(i)} - w^{*T} x^{(i)} \right] = \frac{1}{|\mathbf{S}|} \sum_{i \in \mathbf{S}} \left[t^{(i)} - \sum_{j=1}^{N} \alpha_j^* t^{(j)} \left(x^{(j)} \cdot x^{(i)} \right) \right]$$

- Or equivalently,

$$b^* = \frac{1}{\sum_{i=1}^{N} \delta(\alpha_i > 0)} \sum_{i=1}^{N} \left[t^{(i)} - \sum_{j=1}^{N} \alpha_j^* t^{(j)} \left(x^{(j)} \cdot x^{(i)} \right) \right] \delta(\alpha_i > 0)$$

> where $\delta(\alpha_i > 0)$ is an indicator function, taking value 1 if $\alpha_i > 0$ and 0 otherwise.

What if data is not linearly separable?



- Introduce **slack variables** ξ_i:

$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + \lambda \sum_{i=1}^{N} \xi_i$$

$$\text{subject to} \quad \xi_i \geq 0; \quad t^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i, \quad \forall i = 1, \ldots, N$$

What if data is not linearly separable?

- Optimization with **slack variables** ξ_i:

$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + \lambda \sum_{i=1}^{N} \xi_i$$

$$\text{subject to} \quad \xi_i \geq 0; \quad t^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i, \quad \forall i = 1, \ldots, N$$

- Example lies on wrong side of hyperplane $\xi_i > 1$
- Therefore $\sum_i \xi_i$ upper bounds the number of training errors
- λ trades off training error vs model complexity
- This is known as the **soft-margin** extension

Convert to Unconstrained Optimization

- Define:

$$J(w, b, \xi; \alpha, \mu) = \frac{1}{2}\|w\|^2 + \lambda \sum_{i=1}^{N} \xi_i + \sum_{i=1}^{N} \alpha_i \left[1 - \xi_i - t^{(i)}(w^T x^{(i)} + b)\right] + \sum_{i=1}^{N} \mu_i(-\xi_i)$$

- Then the problem becomes:

$$\max_{\alpha_i \geq 0, \mu_i \geq 0} \min_{w,b,\xi} J(w, b, \xi; \alpha, \mu)$$

- Consdier the KKT conditions:

$$\frac{\partial J}{\partial w} = w - \sum_{i=1}^{N} \alpha_i x^{(i)} t^{(i)} = 0$$

$$\frac{\partial J}{\partial b} = -\sum_{i=1}^{N} \alpha_i t^{(i)} = 0$$

$$\frac{\partial J}{\partial \xi_i} = \lambda - \alpha_i - \mu_i = 0, \quad \forall i = 1, \ldots, N$$

Solving Lagrange Multipliers

- Note that $\forall i = 1, \ldots, N$:

$$\alpha_i \geq 0$$
$$\mu_i \geq 0$$

- We can also obtain:

$$0 \leq \alpha_i = \lambda - \mu_i \leq \lambda$$

- So, considering the KKT condition, we have:

$$\max \left\{ \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} t^{(i)} t^{(j)} \alpha_i \alpha_j \left(x^{(i)} \cdot x^{(j)} \right) \right\}$$
$$\text{subject to} : 0 \leq \alpha_i \leq \lambda, \quad \forall i = 1, \ldots, N$$
$$\sum_{i=1}^{N} \alpha_i t^{(i)} = 0$$

Obtain the Parameters for SVM

- The weights are

$$w = \sum_{i=1}^{N} \alpha_i t^{(i)} x^{(i)}$$

- Only a small subset of α_i's will **satisfy** $0 < \alpha_i < \lambda$, and the corresponding $x^{(i)}$'s are the **support vectors** **S**
- Prediction on a new example:.

$$y = \text{sign}\left[b + x \cdot \left(\sum_{i=1}^{N} \alpha_i t^{(i)} x^{(i)}\right)\right]$$

Obtain the Parameters for SVM

- For support vectors $x^{(i)} \in \mathbf{S}$, we have:

$$1 - (w^T x^{(i)} + b)t^{(i)} = 0$$

- Thus, by using any one support vector $x^{(i)} \in \mathbf{S}$, we can solve b:

> $$b = t^{(i)} - w^T x^{(i)} = t^{(i)} - \sum_{j=1}^{N} \alpha_j t^{(j)} \left(x^{(j)} \cdot x^{(i)} \right)$$

- Or averaging over all support vectors:

> $$b = \frac{1}{|\mathbf{S}|} \sum_{i \in \mathbf{S}} \left[t^{(i)} - w^T x^{(i)} \right] = \frac{1}{|\mathbf{S}|} \sum_{i \in \mathbf{S}} \left[t^{(i)} - \sum_{j=1}^{N} \alpha_j t^{(j)} \left(x^{(j)} \cdot x^{(i)} \right) \right]$$

Let's talk about α_i

- When $\alpha_i = 0$,

$$\lambda - \alpha_i - \mu_i = 0 \;\Rightarrow\; \mu_i = \lambda \;\Rightarrow\; \mu_i > 0$$

$$\left\{ \begin{matrix} \mu_i > 0 \\ \mu_i(-\xi_i) = 0 \end{matrix} \right\} \;\Rightarrow\; \xi_i = 0$$

$$1 - \xi_i - t^{(i)}(w^T x^{(i)} + b) \leq 0 \;\Rightarrow\; t^{(i)}(w^T x^{(i)} + b) \geq 1$$

- When $0 < \alpha_i < \lambda$,

$$\lambda - \alpha_i - \mu_i = 0 \;\Rightarrow\; \mu_i = \lambda - \alpha_i \;\Rightarrow\; \mu_i > 0$$

$$\left\{ \begin{matrix} \mu_i > 0 \\ \mu_i(-\xi_i) = 0 \end{matrix} \right\} \;\Rightarrow\; \xi_i = 0$$

$$1 - \xi_i - t^{(i)}(w^T x^{(i)} + b) = 0 \;\Rightarrow\; t^{(i)}(w^T x^{(i)} + b) = 1$$

Let's talk about α_i (Cont'd)

- When $\alpha_i = \lambda$,

$$\lambda - \alpha_i - \mu_i = 0 \quad \Rightarrow \quad \mu_i = 0$$

$$\left\{ \begin{array}{c} \mu_i = 0 \\ \mu_i(-\xi_i) = 0 \end{array} \right\} \quad \Rightarrow \quad \xi_i \geq 0$$

$$1 - \xi_i - t^{(i)}(w^T x^{(i)} + b) = 0 \quad \Rightarrow \quad t^{(i)}(w^T x^{(i)} + b) \leq 1$$

Summary

1. $\alpha_i = 0 \iff t^{(i)}(w^T x^{(i)} + b) \geq 1$ (sample i is on the correct side with $\xi_i = 0$)
2. $0 < \alpha_i < \lambda \iff t^{(i)}(w^T x^{(i)} + b) = 1$ (sample i is a support vector)
3. $\alpha_i = \lambda \iff t^{(i)}(w^T x^{(i)} + b) \leq 1$ (sample i is on the wrong side with $\xi_i \neq 0$)

SMO Algorithm

- Sovle the following optimization by SMO (**Sequential Minimal Optimization**)

$$\max \left\{ \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} t^{(i)} t^{(j)} \alpha_i \alpha_j \left(x^{(i)} \cdot x^{(j)} \right) \right\}$$
$$\text{subject to} : 0 \leq \alpha_i \leq \lambda, \quad \forall i = 1, \ldots, N$$
$$\sum_{i=1}^{N} \alpha_i t^{(i)} = 0$$

- **Idea:** Select **two variables** α_i **and** α_j at a time by treating others as constants, and solve the corresponding quadratic programming problem
  - Actually, since we have an equality constraint, we only need to solve a quadratic programming problem with **only one variable** α_i

SMO Algorithm

Rough idea:

1. Select two variables, say without loss of generality, α_1 and α_2, and treat all others as constants, that is,

$$\alpha_1 t^{(1)} + \alpha_2 t^{(2)} = -\sum_{i=3}^{N} \alpha_i t^{(i)} \quad \Rightarrow \quad \alpha_1 t^{(1)} + \alpha_2 t^{(2)} = \beta$$

2. Express α_1 in terms of α_2:

$$\alpha_1 = \left(\beta - \alpha_2 t^{(2)}\right) t^{(1)}$$

SMO Algorithm

3. Substitude α_1 into the objective function:

$$\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} t^{(i)} t^{(j)} \alpha_i \alpha_j \left(x^{(i)} \cdot x^{(j)} \right) = a\alpha_2^2 + b\alpha_2 + c$$

> where a, b, c are constants.

4. It is certain that the maximizer of $a\alpha_2^2 + b\alpha_2 + c$ is obtained when α_2 is equal to:

$$\alpha_2 = -\frac{b}{2a}$$

5. Go back to step 1.

Convex Optimization Algorithm

- Equivalent quadratic programming in the matrix form:

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2}\alpha^T Q \alpha + p^T \alpha \\ \text{s.t.} \quad & t^T \alpha = 0 \\ & 0 \le \alpha_i \le \lambda,\ i = 1, \cdots, N \end{aligned}$$

- For this formula,

$$Q(i,j) = t^{(i)} t^{(j)} \left(x^{(i)} \cdot x^{(j)} \right), \quad p(i) = -1$$

- So

$$Q = \left(\begin{bmatrix} t^{(1)} \\ \vdots \\ t^{(N)} \end{bmatrix} \begin{bmatrix} t^{(1)} & \cdots & t^{(N)} \end{bmatrix} \right) \odot \left(\begin{bmatrix} x^{(1)} \\ \vdots \\ x^{(N)} \end{bmatrix} \begin{bmatrix} x^{(1)} & \cdots & x^{(N)} \end{bmatrix} \right) = \left(tt^T \right) \odot \left(XX^T \right)$$

> ❝ $\odot$ denotes element-wise product, also known as **Hadamard product**.

Nonlinear Decsion Boundaries

- Note that both the learning objective and the decision function depend only on dot products between patterns

$$\begin{aligned} \ell &= \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} t^{(i)} t^{(j)} \alpha_i \alpha_j \left(x^{(i)} \cdot x^{(j)} \right) \\ y &= \text{sign} \left[b + x \cdot \left(\sum_{i=1}^{N} \alpha_i t^{(i)} x^{(i)} \right) \right] \end{aligned}$$

- **How to form non-linear decision boundaries in input space?**

Nonlinear Decsion Boundaries

- **How to form non-linear decision boundaries in input space?**

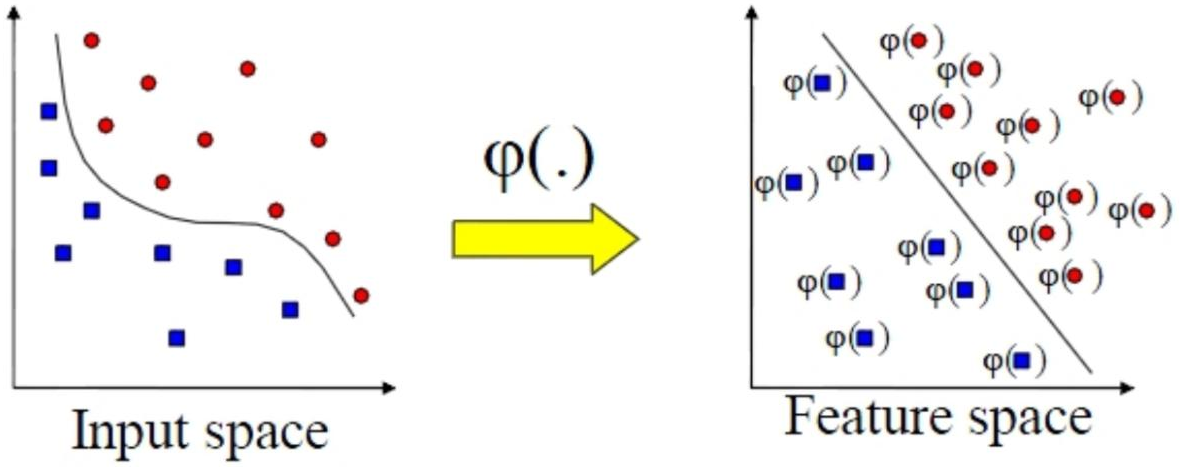
1. Map data into feature space $x \rightarrow \phi(x)$
2. Replace dot products between inputs with feature points

$$x^{(i)} \cdot x^{(j)} \rightarrow \phi(x^{(i)}) \cdot \phi(x^{(j)})$$

3. Find linear decision boundary in feature space

- **Problem:** What is a good feature function $\phi(x)$?

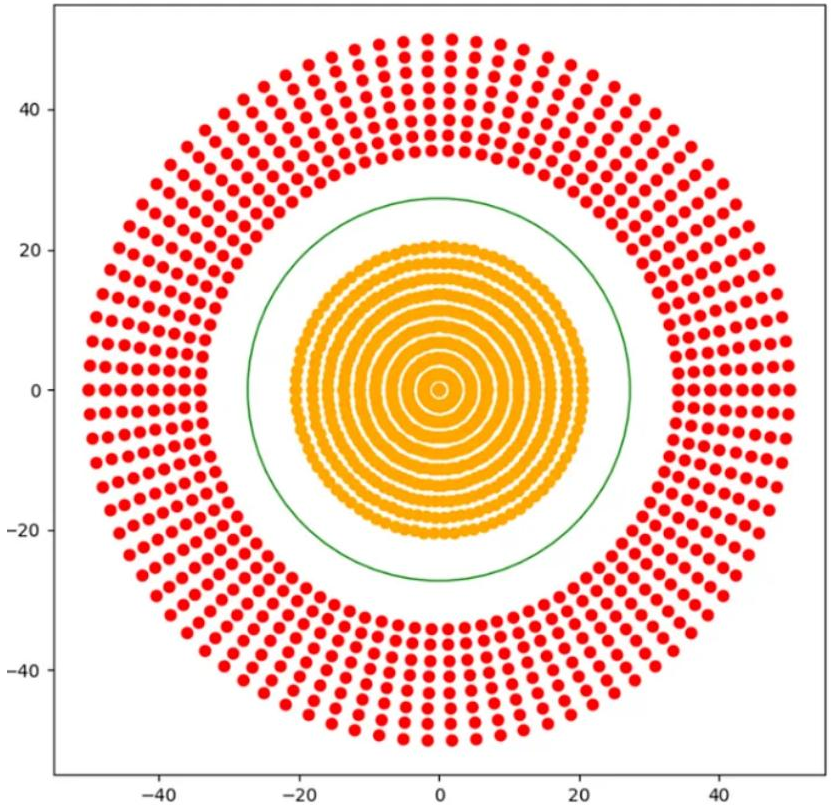
Input Transformation



- Mapping to a feature space can produce problems:
  - High computational burden due to high dimensionality
  - Many more parameters

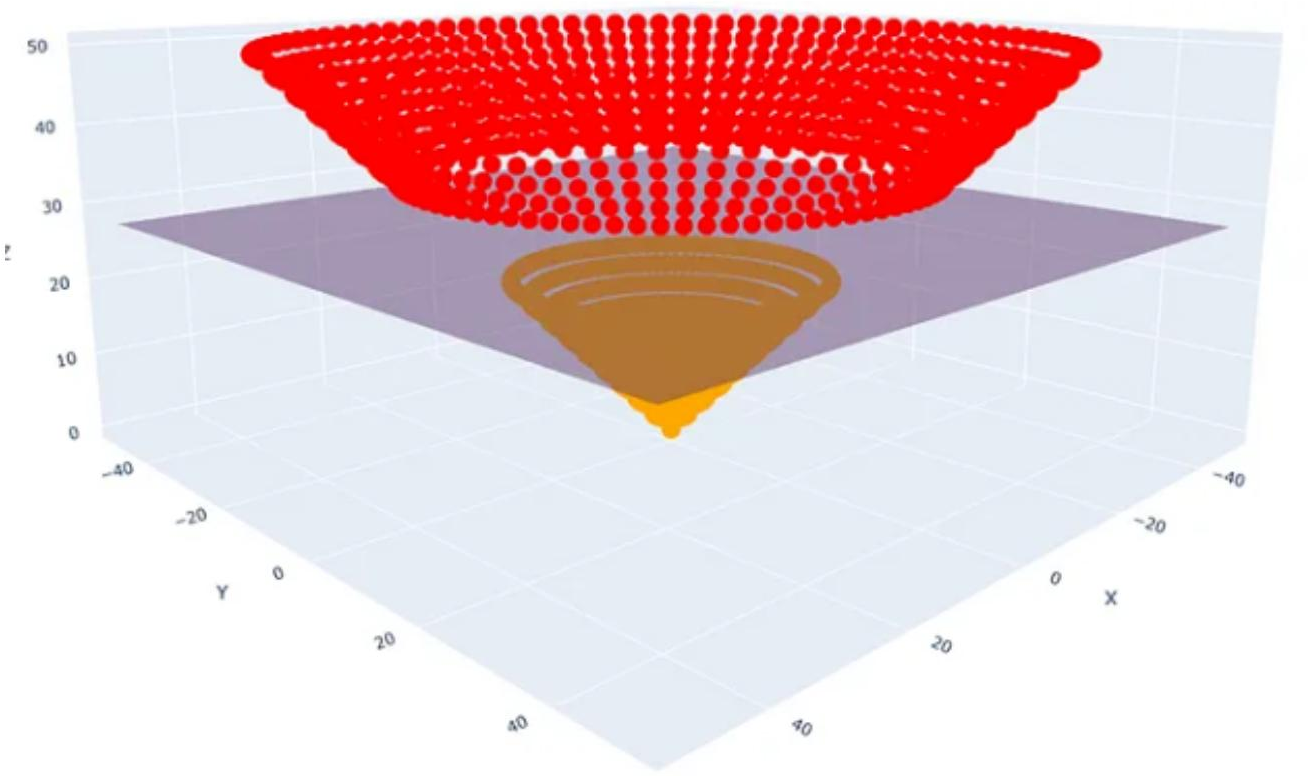
Not Linearly Seperable in the Feature Space

- In the original feature space, the data is not linearly separable
- Each data is of 2-dimension, namely (x, y)



Linearly Seperable in Higher-Dim Space

- Map to a higher dimensional space, the data becomes linearly separable
- Each data now is of 3-dimension, namely $(x, y, \sqrt{x^2 + y^2})$



Kernel Tricks

- SVM solves these two issues simultaneously
  - "Kernel trick" produces efficient classification
  - Dual formulation only assigns parameters to samples, not features
- **Kernel trick**: dot-products in feature space can be computed as a **kernel function**

$$K(x^{(i)}, x^{(j)}) = \phi(x^{(i)}) \cdot \phi(x^{(j)})$$

- **Idea**: work directly on x, avoid having to compute $\phi(x)$

Kernel Tricks

- Example:

$$
\begin{aligned}
K(a,b) &= (a^T b)^3 \\
&= ((a_1,a_2)^T(b_1,b_2))^3 \\
&= (a_1b_1 + a_2b_2)^3 \\
&= a_1^3b_1^3 + 3a_1^2b_1^2a_2b_2 + 3a_1b_1a_2^2b_2^2 + a_2^3b_2^3 \\
&= (a_1^3, \sqrt{3}a_1^2a_2, \sqrt{3}a_1a_2^2, a_2^3)^T(b_1^3, \sqrt{3}b_1^2b_2, \sqrt{3}b_1b_2^2, b_2^3) \\
&= \phi(a) \cdot \phi(b)
\end{aligned}
$$

Kernels

- Examples of kernels: **kernels measure similarity**.

1. Polynomial

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

> where d is the degree of the polynomial, e.g, $d = 2$ for quadratic

2. Gaussian

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

3. Sigmoid

$$K(x^{(i)}, x^{(j)}) = \tanh(\beta(x^{(i)^T} x^{(j)}) + a)$$

Kernels

- Each kernel computation corresponds to dot product
  - Calculation for particular mapping $\phi(x)$ implicitly maps to high-dimensional space
- Why is this useful?
  i. Rewrite training examples using more complex features
  ii. Dataset not linearly separable in original space may be linearly separable in higher dimensional space

Kernel Functions

- **Mercer's Theorem** (1909): Any reasonable kernel corresponds to some feature space
- Reasonable means that the **Gram matrix** is **positive definite**

$$K_{ij} = K(x^{(i)}, x^{(j)})$$

- Feature space can be very large
  - Polynomial kernel $\left(1 + (x^{(i)})^T x^{(j)}\right)^d$ corresponds to feature space exponential in d
  - Gaussian kernel has infinitely dimensional features
- Linear separators in these super high-dim spaces correspond to **highly nonlinear decision boundaries** in input space

Classification with Non-linear SVMs

- Non-linear SVM using kernel function $K()$:

$$\ell = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} t^{(i)} t^{(j)} \alpha_i \alpha_j K(x^{(i)}, x^{(j)})$$

- Maximize ℓ w.r.t. $\{\alpha_i\}$ under constraints $\alpha_i \geq 0, \forall i = 1, \ldots, N$.

$$\max \left\{ \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} t^{(i)} t^{(j)} \alpha_i \alpha_j K(x^{(i)}, x^{(j)}) \right\}$$

$$\text{subject to} : 0 \leq \alpha_i \leq \lambda, \quad \forall i = 1, \ldots, N$$

$$\sum_{i=1}^{N} \alpha_i t^{(i)} = 0$$

Convex Optimization Algorithm

- Equivalent quadratic programming in the matrix form:

$$\begin{aligned} \min_{\alpha} \quad & \tfrac{1}{2}\alpha^T Q\alpha + p^T\alpha \\ \text{s.t.} \quad & t^T\alpha = 0 \\ & 0 \le \alpha_i \le \lambda,\ i = 1, \cdots, N \end{aligned}$$

- For this formula,

$$Q(i,j) = t^{(i)}t^{(j)}K\left(x^{(i)}, x^{(j)}\right), \quad p(i) = -1$$

- So

$$Q = \left(\begin{bmatrix} t^{(1)} \\ \vdots \\ t^{(N)} \end{bmatrix} \begin{bmatrix} t^{(1)} & \cdots & t^{(N)} \end{bmatrix}\right) \odot \left(\begin{bmatrix} x^{(1)} \\ \vdots \\ x^{(N)} \end{bmatrix} \begin{bmatrix} x^{(1)} & \cdots & x^{(N)} \end{bmatrix}\right) = \left(tt^T\right) \odot K$$

> K is the **Gram matrix** with $K_{ij} = K(x^{(i)}, x^{(j)})$
>
> $\odot$ denotes element-wise product, also known as **Hadamard product**.

Obtain the Parameters for SVM

- The weights **for linear SVM** are

$$w = \sum_{i=1}^{N} \alpha_i t^{(i)} x^{(i)}$$

- Unlike linear SVM, **nonlinear SVM cannot express** w as linear combination of support vectors
- But, it does not matter as the prediction **does not need to know** w exactly.

Prediction using SVM

- Prediction on a new example with **linear SVM**:.

$$y = \text{sign}\left[b + x \cdot \left(\sum_{i=1}^{N} \alpha_i t^{(i)} x^{(i)}\right)\right]$$

- Prediction on a new example with **nonlinear SVM**:.

$$y = \text{sign}\left[b + \sum_{i=1}^{N} t^{(i)} \alpha_i K\left(x, x^{(i)}\right)\right]$$

- In the prediction formula, it suffices to have kernel function, but b should be known.

Obtain the Parameters for SVM

- Linear SVM

$$b = t^{(i)} - w^T x^{(i)} = t^{(i)} - \sum_{j=1}^{N} \alpha_j t^{(j)} \left(x^{(j)} \cdot x^{(i)} \right)$$

- Similarly, we have b as follows for nonlinear SVM:

$$b = t^{(i)} - \sum_{j=1}^{N} \alpha_j t^{(j)} K \left(x^{(j)}, x^{(i)} \right)$$

> where $x^{(i)}$ is a support vector.

Advantages and Disadvantages

- Advantages:
  - Kernels allow very flexible hypotheses
  - Poly-time exact optimization methods rather than approximate methods
  - Soft-margin extensions permits mis-classified examples
  - Excellent results (1.1% error rate on handwritten digits vs. LeNet's 0.9%)
- Disadvantages:
  - Must choose kernel parameters

More Summary

- Difference between logistic regression and SVMs
- Maximum margin principle
- Target function for SVMs
- Slack variables for mis-classified points
- Kernel trick allows non-linear generalizations