

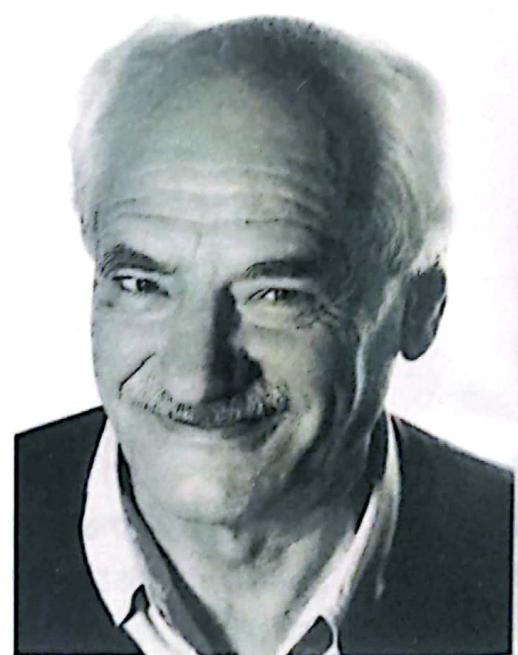
অধ্যায় ৫ : সাপোর্ট ভেস্ট্র মেশিন (Support Vector Machine - SVM)

সাপোর্ট ভেস্ট্র মেশিন অ্যালগরিদমটি প্রথম আবিষ্কার করেন ভ্লাদিমির (Vladimir Vapnik 1936) ও আলেক্সেই (Alexey Chervonenkis, 1938 - 2014) নামের দুজন বিজ্ঞানী, ১৯৬৩ সালে।

পরবর্তী সময়ে বোসের (Bernhard E. Boser), ইসাবেল (Isabelle M. Guyon) ও ভ্লাদিমির মিলে ১৯৯২ সালে আধুনিক সাপোর্ট ভেস্ট্র মেশিন ও কার্নাল ট্রিক সম্পর্কে ধারণা দেন, যা কলিনিয়ার ডেটা ক্লাসিফাই করার জন্য পরবর্তীতে ব্যবহৃত হয়।



Vladimir Vapnik, 1936

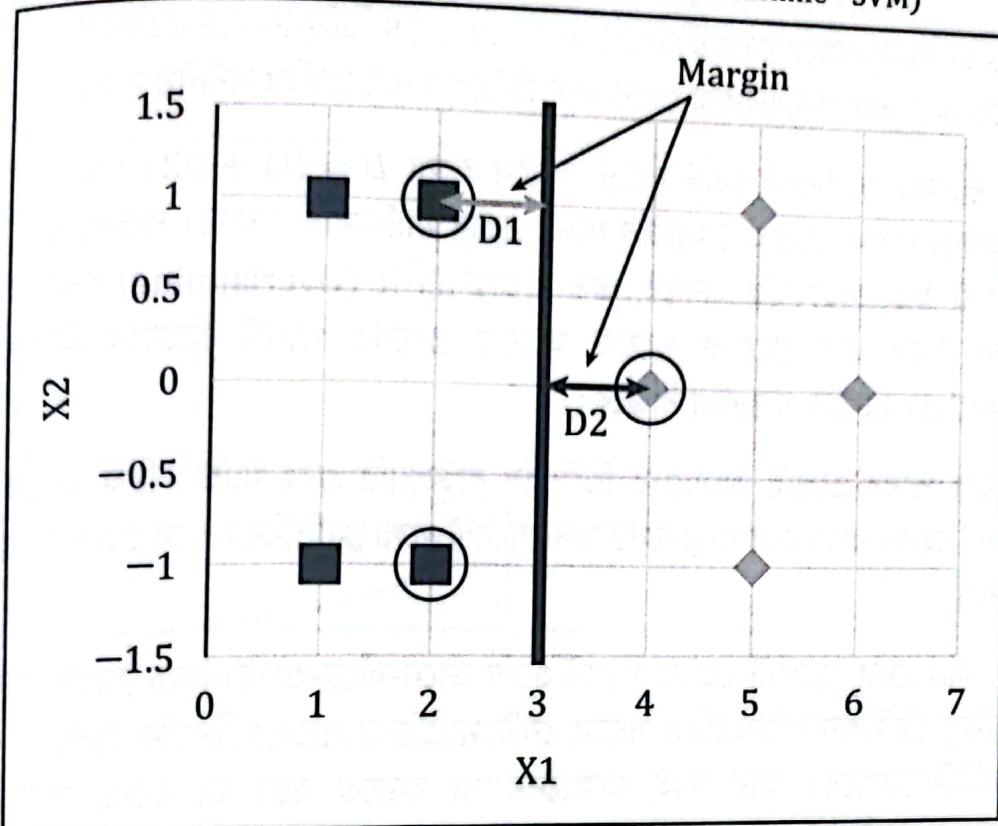


Alexey Chervonenkis, 1938 - 2014

ছবি 5.1

সাপোর্ট ভেস্ট্র মেশিন খুব সন্তুষ্ট মেশিন লার্নিংয়ের সবচেয়ে বহুল ব্যবহৃত এবং অন্যতম জনপ্রিয় অ্যালগরিদম। এই অ্যালগরিদম বহু জায়গায়, বহু প্রবলেমসেটে বহুভাবে ব্যবহৃত হয়েছে। এ জন্য অসংখ্য অপটিমাইজেশন মেথড রয়েছে, সেই সঙ্গে রয়েছে এর নানাবিধ অ্যাপ্লিকেশন।

সাপোর্ট ভেস্ট্র মেশিনের উদাহরণসহ ব্যবহারের পদ্ধতি শেখার আগে তাই আমরা এর গঠন সম্পর্কে প্রথমে একটুখানি জেনে নেব।



ছবি 5.2

ওপরের গ্রাফটি (ছবি 5.2) লক্ষ করি। ধরি, বর্গাকৃতির চিহ্ন দিয়ে দেখানো চারটি পয়েন্ট Class 1-এর এবং ডায়মন্ড আকৃতির চিহ্ন দিয়ে দেখানো চারটি পয়েন্ট Class 2-এর।

এখন, যদি আমরা Class 1-এর প্রতিটি পয়েন্টের সঙ্গে Class 2-এর প্রতিটি পয়েন্টের দূরত্ব বের করি এবং সেখান থেকে বের করার চেষ্টা করি যে, এই দুই ক্লাসের কোন কোন পয়েন্টগুলো একে অপরের সবচেয়ে কাছাকাছি আছে (যে দুটো পয়েন্টের মধ্যে দূরত্ব মাপব, তারা অবশ্যই ভিন্ন ভিন্ন ক্লাসের হতে হবে), তখন দেখব যে $(2, 1), (2, -1)$ ও $(4, 0)$ – এই তিনটি পয়েন্ট একে অপরের সবচেয়ে কাছাকাছি (গ্রাফে বৃত্তাকৃতির চিহ্ন দিয়ে দেখানো হয়েছে)।

আরো লক্ষ করি, আমি যদি কোনো একটি রেখা টেনে এই দুটি ক্লাসকে আলাদা করতে চাই (মাঝখানে মোটা দাগ দিয়ে দেখানো, একে Decision Boundary বলে), তবে দেখুন যে $(2, 1), (2, -1)$ ও $(4, 0)$ এই তিনটি পয়েন্টই ডিসিশন বাউন্ডারির সবচেয়ে কাছে থাকবে।

আর তাই এই পয়েন্টগুলোকে বলা হয় সাপোর্ট ভেক্টর। গ্রাফটি ভালো করে লক্ষ করলে দেখা যাবে যে, $(2, 1)$ ও $(2, -1)$ হচ্ছে ডিসিশন বাউন্ডারি থেকে Class 1-এর সবচেয়ে কাছের দুটি পয়েন্ট এবং এদের থেকে ডিসিশন বাউন্ডারির দূরত্ব D_1 । তাই বলা যায়, Class 1-এর পয়েন্টগুলো থেকে এই ডিসিশন বাউন্ডারির সর্বনিম্ন দূরত্ব D_1 । একইভাবে, $(4, 0)$ হচ্ছে ডিসিশন বাউন্ডারি

থেকে Class 2-এর সবচেয়ে কাছের পয়েন্ট এবং এর থেকে ডিসিশন বাউন্ডারির দূরত্ব D₂। তাই
বলা যায়, Class 2-এর পয়েন্টগুলো থেকে এই ডিসিশন বাউন্ডারির সর্বনিম্ন দূরত্ব D₂।

এখন, দুটি ক্লাসের মধ্যেকার মোট দূরত্ব তাহলে হলো $D = D_1 + D_2$ । একে আমরা বৃদ্ধি মার্জিন। আমাদের লক্ষ্য হচ্ছে এই মার্জিন যতটা সম্ভব ম্যাগ্নিমাইজ করা। কারণ, দুটি ডিসিশন ক্লাসের পয়েন্ট যত কাছাকাছি থাকবে, তত ওভারল্যাপিং (Overlapping) হওয়ার সম্ভাবনা বাঢ়বে এবং ফলে এক ক্লাসের পয়েন্ট আরেক ক্লাসের পয়েন্ট হিসেবে মিসক্লাসিফাই (Misclassified) হওয়ার সম্ভাবনা বাঢ়বে।

সাপোর্ট ভেক্টর পয়েন্টগুলোই আমাদের ডিসিশন বাউন্ডারি এবং মার্জিন ঠিক করতে সহায় করে। সাপোর্ট ভেক্টর বাদে কোনো ক্লাসের অন্য পয়েন্টগুলো ক্লাসিফিকেশনে তেমন কোনো উপর বহন করে না।

আমরা যদি আমাদের সাপোর্ট ভেক্টর পয়েন্টগুলো ডানে-বাঁয়ে-ওপরে-নিচে সরাই, তাহলে তাৰ সঙ্গে সঙ্গে কিন্তু ডিসিশন বাউন্ডারিও সরবে, যেটি মার্জিনের মানে পরিবর্তন আনবে এবং প্রভাব ফেলবে ক্লাসিফিকেশনে। আর তাই আমাদের যা করতে হবে তা হলো, সাপোর্ট ভেক্টর পয়েন্টগুলোর মধ্যবর্তী এমন কোনো জায়গায় ডিসিশন বাউন্ডারি আঁকতে হবে, যাতে মার্জিন বৃদ্ধি পায়। এই ডিসিশন বাউন্ডারি খুঁজে বের করার কাজই করে দেয় সাপোর্ট ভেক্টর মেশিন অ্যালগরিদম। সাপোর্ট ভেক্টর মেশিন সব সময় বহুম মার্জিন (Largest Margin) খুঁজে বের করার চেষ্টা করে দেখে একে লার্জেস্ট/ম্যাক্সিমাম মার্জিন ক্লাসিফায়ার (Largest/Maximum Margin Classifier)-ও বলা হয়।

পরিচ্ছেদ ৫.১ : লিনিয়ার সাপোর্ট ভেক্টর মেশিন (Linear Support Vector Machine - LSVM)

আমরা লজিস্টিক রিপ্রেশন পড়ার সময় হাইপোথিসিস হিসেবে নিয়েছিলাম –

$$h_w(x) = g(z) = \frac{1}{1+e^{-z}}$$

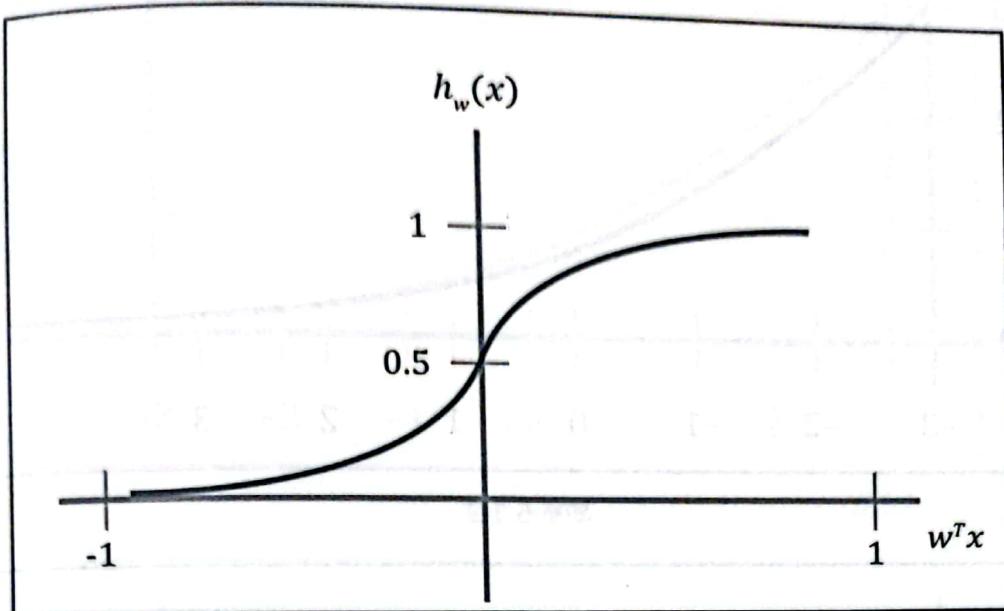
যেখানে, $z = W^T x$ ছিল। একে যদি আমরা একটি গ্রাফে প্লট করি, তাহলে গ্রাফ 5.1.1-এর মত একটি গ্রাফ পাব।

লজিস্টিক রিপ্রেশনের ক্ষেত্রে,

- ✓ যদি প্রকৃত আউটপুট $y = 1$ হয়, তাহলে আমরা চাই $h_w(x) \approx 1$ হোক, অর্থাৎ $W^T x \gg 0$ হোক। আবার,

অধ্যায় ৫ : সাপোর্ট ভেস্ট্র মেশিন (Support Vector Machine - SVM)

- যদি প্রকৃত আউটপুট $y = 0$ হয়, তাহলে আমরা চাই $h_w(x) \approx 0$ হোক, অর্থাৎ $W^T x \ll 0$ হোক।



গ্রাফ 5.1.1

এখন, আমরা যা করব তা হলো, লজিস্টিক রিগ্রেশনে যা আমরা পড়েছি, তাকেই কিছুটা পরিবর্তিত করে সেখান থেকে কীভাবে সাপোর্ট ভেস্ট্র মেশিন শেখা যায় তা দেখব।

লজিস্টিক রিগ্রেশনের ক্ষেত্রে, একটিমাত্র ডেটা পয়েন্টের জন্য Cost ছিল –

$$Cost = -y \log(h_w(x)) - (1 - y) \log(1 - h_w(x))$$

এখন, এই Cost ব্যবহার করে আমরা ওপরে যে দুটি প্রেক্ষাপট উল্লেখ করলাম ($y = 1$ ও $y = 0$) সেই দুটি নিয়ে একটু বিস্তারিত বিশ্লেষণ করি –

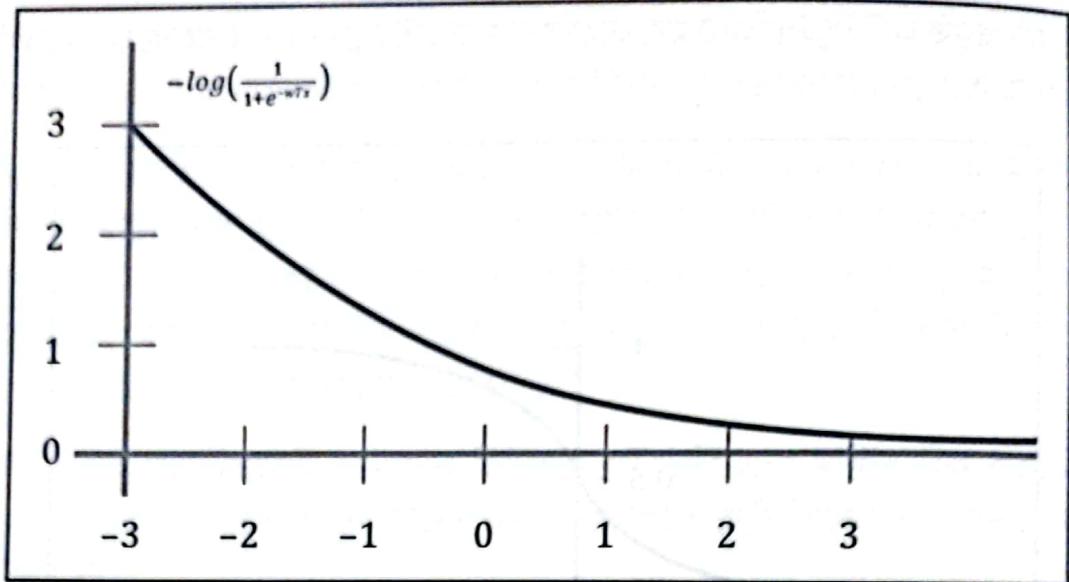
আমরা যদি, $y = 1$ এর জন্য, ($W^T x \gg 0$) নিই, সে ক্ষেত্রে আমাদের কস্ট দাঁড়ায়,

$$Cost = -\log(h_w(x)) = -\log\left(\frac{1}{1+e^{-W^T x}}\right)$$

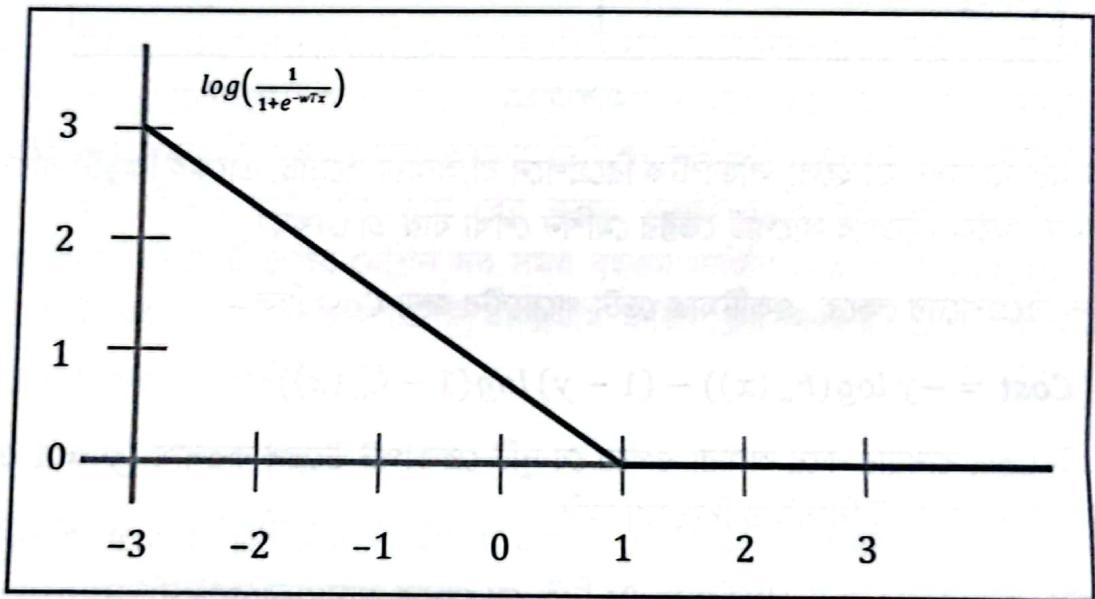
এখন, শুধু এই অংশটুকু কে যদি আমি গ্রাফে প্লট করি, তাহলে গ্রাফ 5.1.2-এর মত একটি গ্রাফ পাব।

এখন সাপোর্ট ভেস্ট্র মেশিন তৈরির জন্য, আমরা গ্রাফ 5.1.2-এর কার্ডটিকে দুই ভাগে ভাগ করব গ্রাফ 5.1.3-এর মতো।

এই গ্রাফ 5.1.3-এর সঙ্গে গ্রাফ 5.1.2-এর কার্ডের অনেকখানিই মিল আছে। শুধু পার্থক্য হলো – আগের গ্রাফের চেয়ে এই গ্রাফ একটু বেশি তীক্ষ্ণ।



গ্রাফ 5.1.2



গ্রাফ 5.1.3

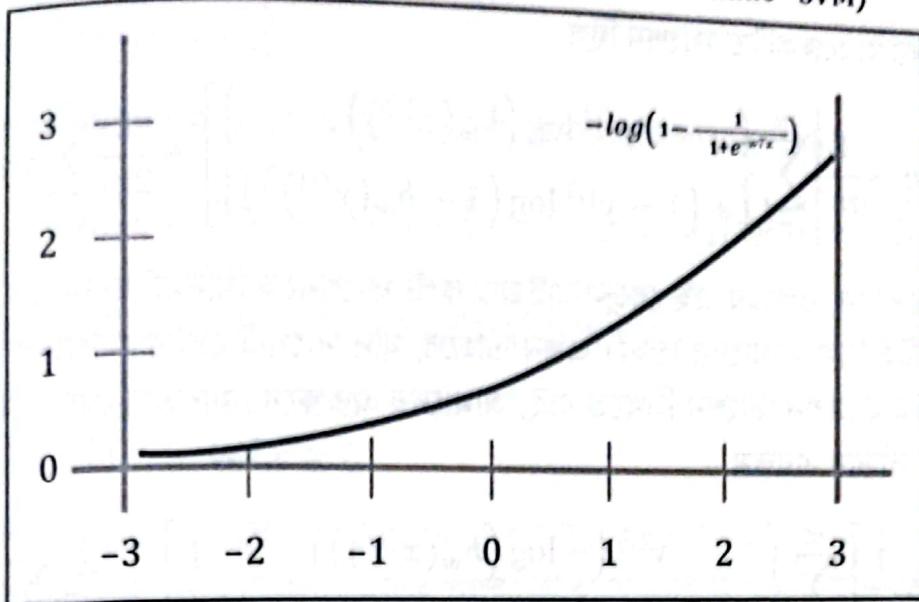
ঠিক একইভাবে, আমরা যদি $y = 0$ এর জন্য ($W^T x \ll 0$) নিই, সে ক্ষেত্রে আমাদের কস্ট দাঁড়ায়,

$$Cost = -\log(1 - h_w(x)) = -\log\left(1 - \frac{1}{1+e^{-W^T x}}\right)$$

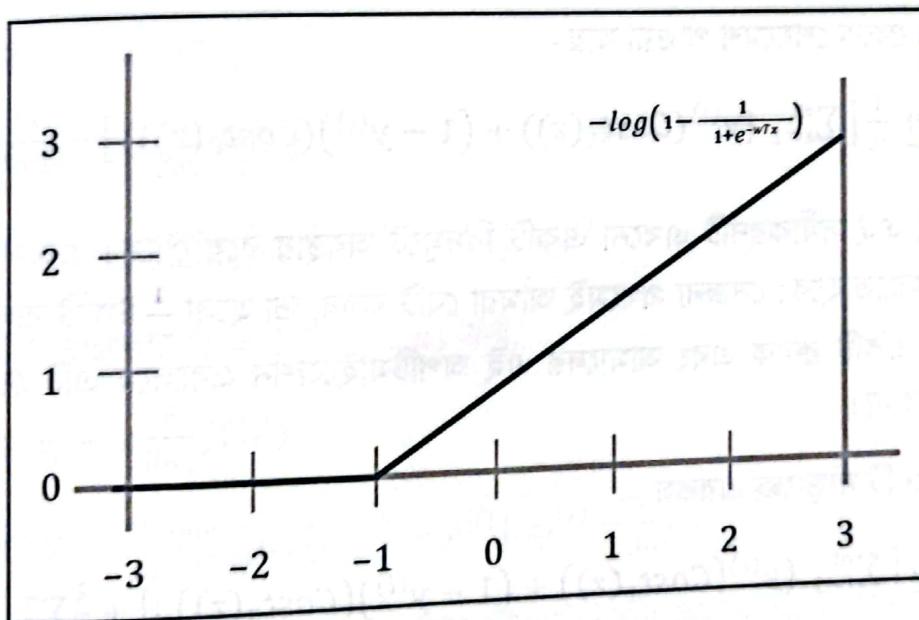
এখন, শুধু এই অংশটুকু যদি আমি গ্রাফে প্লট করি, তাহলে তাহলে গ্রাফ 5.1.4-এর মত একটি গ্রাফ পাব।

এখন আবার আগের মতোই আমরা গ্রাফ 5.1.4-এর কার্ডটিকে দুই ভাগে ভাগ করব গ্রাফ 5.1.5-এর মতো।

অধ্যায় ৫ : সাপোর্ট ভেক্টর মেশিন (Support Vector Machine - SVM)



গ্রাফ 5.1.4



গ্রাফ 5.1.5

এখন, $(y = 1)$ -এর জন্য আমরা যে কস্ট পেলাম, ধরি তা হলো $Cost_1(z)$ ও $(y = 0)$ -এর জন্য যে কস্ট পেলাম, ধরি তা হলো $Cost_0(z)$ ।

পরিচেদ 8.8 থেকে হয়তো সবার মনে আছে যে, আমরা লজিস্টিক রিগ্রেশনের জন্য রেণ্ডুরাইজেশন ব্যবহার করে যে কস্ট ফাংশন পেয়েছিলাম তা ছিল এরকম:

$$J(W) = -\frac{1}{m} \left[\sum_{i=1}^m \left\{ y^{(i)} \log(h_w(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_w(x^{(i)})) \right\} \right] + \frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$

এবং আমাদের অবজেকটিভ ফাংশন ছিল -

$$\min_w -\frac{1}{m} \left[\sum_{i=1}^m \left\{ y^{(i)} \log(h_w(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_w(x^{(i)})) \right\} \right] + \frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$

অর্থাৎ, আমাদেরকে ওপরের এই রেজুলারাইজড কস্ট ফাংশনকে মিনিমাইজ করতে হবে W -এর সাপেক্ষে, এটি ছিল আমাদের লক্ষ্য। এখন তাহলে, যদি সাপোর্ট ভেষ্টের মেশিনের জন্য এরকম একটি অপটিমাইজেশন ফাংশন লিখতে যাই, আমাদের এতক্ষণের যাবতীয় কাজকর্মের সাপেক্ষে তাহলে সেটি দাঁড়ায় এরকম -

$$\min_w \frac{1}{m} \left[\sum_{i=1}^m \left\{ y^{(i)} (-\log(h_w(x^{(i)}))) + (1 - y^{(i)}) (-\log(1 - h_w(x^{(i)}))) \right\} \right] + \frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$

এবং সেখান থেকে শেষমেশ পাওয়া যায় -

$$\min_w \frac{1}{m} [\sum_{i=1}^m \{y^{(i)}(Cost_1(z)) + (1 - y^{(i)})(Cost_0(z))\}] + \frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$

এখন দেখুন, এই সমীকরণটি এখনো একটি বিদ্যুটে অবস্থায় রয়ে গেছে। একে আরো একটি সরলীকরণ করতে হবে। সেজন্য প্রথমেই আমরা যেটি করব, তা হলো $\frac{1}{m}$ টার্মটি বাদ দিয়ে দেব কেননা এটি একটি ধ্রুবক এবং আমাদের এই অপটিমাইজেশন প্রবলেমে এটি তেমন কোনো প্রভাব ফেলবে না।

তাহলে ফাংশনটি দাঁড়াচ্ছে এরকম -

$$\min_w [\sum_{i=1}^m \{y^{(i)}(Cost_1(z)) + (1 - y^{(i)})(Cost_0(z))\}] + \frac{\lambda}{2} \sum_{j=1}^n w_j^2$$

এখন এই ফাংশনটি আমরা এইভাবে লিখতে পারি -

$$A + \lambda B$$

যেখানে, $A = \min_w [\sum_{i=1}^m \{y^{(i)}(Cost_1(z)) + (1 - y^{(i)})(Cost_0(z))\}]$ এবং

$$B = \frac{1}{2} \sum_{j=1}^n w_j^2$$

এখন, ওপরের এই $A + \lambda B$ -কে আমরা একটু বদলে এইভাবেও লিখতে পারি,

অধ্যায় ৫ : সাপোর্ট ভেস্ট্র মেশিন (Support Vector Machine - SVM)

$$CA + B \text{ যেখানে, } C = \frac{1}{\lambda}$$

সুতরাং সবশেষে আমাদের ফাইনাল অপটিমাইজেশন প্রবলেমটি দাঁড়ায় এরকম –

$$\min_w C \left[\sum_{i=1}^m \{y^{(i)}(Cost_1(z)) + (1 - y^{(i)})(Cost_0(z))\} \right] + \frac{1}{2} \sum_{j=1}^n w_j^2$$

এবং আমাদের সাপোর্ট ভেস্ট্র মেশিনের জন্য হাইপোথিসিস হবে –

$$h_w(x) = \begin{cases} 0, & W^T x < 0 \\ 1, & W^T x \geq 0 \end{cases}$$

এখন আসি প্যারামিটার কীভাবে আপডেট করব সে আলোচনায়। এটিও আগের পরিচ্ছেদে দেখানো লজিস্টিক রিগ্রেশনের রেগুলারাইজড প্যারামিটার যেভাবে আপডেট করা হচ্ছে, সেভাবে করতে হবে –

প্রথম প্যারামিটার w_0 -এর জন্য,

$$w_0 = w_0 - \alpha \frac{\partial}{\partial w_0} J(W)$$

$$\frac{\partial}{\partial w_0} J(W) = \frac{1}{m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\} \cdot x_0^{(i)}$$

পরবর্তী প্যারামিটারগুলোর ($j = 1, 2, 3, \dots, n$) জন্য,

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(W)$$

$$\frac{\partial}{\partial w_j} J(W) = \frac{1}{m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\} \cdot x_j^{(i)} - \frac{\lambda}{m} w_j$$

এই হলো একটি লিনিয়ার সাপোর্ট ভেস্ট্র মেশিন কীভাবে তৈরি করতে হয় তার বর্ণনা। পরবর্তী পরিচ্ছেদে আমরা দেখব কীভাবে, এত জটিল গাণিতিক হিসাবকিতাব একটু এড়িয়ে সহজ করে একটি লিনিয়ার সাপোর্ট ভেস্ট্র মেশিন তৈরি করা যায়।

পরিচ্ছেদ ৫.২ : লিনিয়ার সাপোর্ট ভেস্ট্র মেশিনের একটি সহজ উদাহরণ

আমরা এতক্ষণ যে উদাহরণটি ব্যবহার করলাম, সেটিই আমরা আবার ব্যবহার করব। আমরা ইতিমধ্যেই দেখেছি যে, উদাহরণে আমাদের সাপোর্ট ভেস্ট্র হচ্ছে তিনটি। এদেরকে S_1, S_2 ও S_3

নাম নিই। সেই সঙ্গে, Class 1-কে সার্ভিচিঙ ক্লাস এবং Class 2-কে পার্সিচিঙ ক্লাস করি।

$$\text{সুতরাং, } S_1 = \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix}, S_2 = \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} \text{ এবং } S_3 = \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix}$$

আমরা, এই তিনটি সাপোর্ট ভেক্টরের প্রতিটির সঙ্গে '1' অগ্রমেট (Augment) করল (পৃষ্ঠা ৫) করে রো যোগ করব এবং মান হবে 1) বায়াস (Bias) হিসেবে। এর গাণিতিক কিছু ধরণ আমরা পরবর্তী সময়ে সেটি নিয়ে আরো বিস্তৃতভাবে আলোচনা করল। আপাতত, '1' অগ্রমেট নতুন সাপোর্ট ভেক্টরগুলো হবে –

$$\tilde{S}_1 = \begin{pmatrix} 2 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \tilde{S}_2 = \begin{pmatrix} 2 \\ -1 \\ 1 \\ 1 \end{pmatrix} \text{ এবং } \tilde{S}_3 = \begin{pmatrix} 4 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

এখন, যেহেতু আমাদের সাপোর্ট ভেক্টর তিনটি, তাই আমাদের তিনটি প্যারামিটার পাওয়া এবং এদেরকে আমরা α_1, α_2 ও α_3 দিয়ে চিহ্নিত করব, যারা যথাক্রমে \tilde{S}_1, \tilde{S}_2 ও \tilde{S}_3 -এর সঙ্গে সম্পর্ক আয়োজন করতে হবে। এখন এই তিনটি প্যারামিটারের মান বের করার জন্য আমাদের তিনটি লিনিয়ার ইকুয়েশন সম্পর্ক করতে হবে –

$$\alpha_1 \cdot \tilde{S}_1 \cdot \tilde{S}_1 + \alpha_2 \cdot \tilde{S}_2 \cdot \tilde{S}_1 + \alpha_3 \cdot \tilde{S}_3 \cdot \tilde{S}_1 = -1 \quad (\tilde{S}_1 \text{ Negative Class Point})$$

$$\alpha_1 \cdot \tilde{S}_1 \cdot \tilde{S}_2 + \alpha_2 \cdot \tilde{S}_2 \cdot \tilde{S}_2 + \alpha_3 \cdot \tilde{S}_3 \cdot \tilde{S}_2 = -1 \quad (\tilde{S}_2 \text{ Negative Class Point})$$

$$\alpha_1 \cdot \tilde{S}_1 \cdot \tilde{S}_3 + \alpha_2 \cdot \tilde{S}_2 \cdot \tilde{S}_3 + \alpha_3 \cdot \tilde{S}_3 \cdot \tilde{S}_3 = +1 \quad (\tilde{S}_3 \text{ Positive Class Point})$$

এই সমীকরণগুলো সমাধান করে আমাদেরকে তিনটি প্যারামিটার α_1, α_2 ও α_3 -এর মান করতে হবে। এখন সমীকরণগুলোতে \tilde{S}_1, \tilde{S}_2 ও \tilde{S}_3 -এর মান বসিয়ে পাই –

$$\alpha_1 \begin{pmatrix} 2 \\ 1 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ 1 \\ 1 \\ 1 \end{pmatrix} + \alpha_2 \cdot \begin{pmatrix} 2 \\ -1 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ 1 \\ 1 \\ 1 \end{pmatrix} + \alpha_3 \cdot \begin{pmatrix} 4 \\ 0 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ 1 \\ 1 \\ 1 \end{pmatrix} = -1$$

$$\alpha_1 \cdot \begin{pmatrix} 2 \\ 1 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ -1 \\ 1 \\ 1 \end{pmatrix} + \alpha_2 \cdot \begin{pmatrix} 2 \\ -1 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ -1 \\ 1 \\ 1 \end{pmatrix} + \alpha_3 \cdot \begin{pmatrix} 4 \\ 0 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ -1 \\ 1 \\ 1 \end{pmatrix} = -1$$

$$\alpha_1 \cdot \begin{pmatrix} 2 \\ 1 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 4 \\ 0 \\ 1 \\ 1 \end{pmatrix} + \alpha_2 \cdot \begin{pmatrix} 2 \\ -1 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 4 \\ 0 \\ 1 \\ 1 \end{pmatrix} + \alpha_3 \cdot \begin{pmatrix} 4 \\ 0 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 4 \\ 0 \\ 1 \\ 1 \end{pmatrix} = +1$$

সমীকরণগুলো সরল করে পাই –

অধ্যায় ৫ : সাপোর্ট ভেস্টার মেশিন (Support Vector Machine - SVM)

$$6\alpha_1 + 4\alpha_2 + 9\alpha_3 = -1$$

$$4\alpha_1 + 6\alpha_2 + 9\alpha_3 = -1$$

$$9\alpha_1 + 9\alpha_2 + 17\alpha_3 = +1$$

এখানে সরলীকৃণের জন্য সাধারণ ভেস্টার ডট গুণন পদ্ধতি প্রয়োগ করা হয়েছে। আমি ধরে নিছি, সবাই উচ্চ মাধ্যমিক পর্যায়ে ভেস্টারের ডট গুণন পড়েছেন। আমি তা-ও এখানে আরেকবার সেটি লিখে দিচ্ছি –

$$\text{ধরি, } \vec{A} \text{ ও } \vec{B} \text{ দুটি ভেস্টার এবং } \vec{A} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \text{ এবং } \vec{B} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

$$\text{তাহলে ভেস্টারব্যয়ের ডট গুণন, } \vec{A} \cdot \vec{B} = a_1b_1 + a_2b_2 + a_3b_3$$

সুতরাং এখন ওপরের তিনটি সমীকরণ সমাধান করে পাই, $\alpha_1, \alpha_2 = -3.25$ এবং $\alpha_3 = 3.5$ ।

আমরা আমাদের সব প্যারামিটারের মান পেয়ে গিয়েছি। এখন, এগুলো ব্যবহার করে আমাদেরকে আমাদের ডিসিশন বাউন্ডারি হিসেবে ব্যবহৃত হাইপার প্লেন (Hyper Plane)-টির সমীকরণ বের করতে হবে। এটি বের করতে পারলেই আমাদের কাজ শেষ।

আমাদের হাইপার প্লেনের সমীকরণকে যদি $y = wx + b$ ধরি, যেখানে b হচ্ছে বায়াস এবং W হচ্ছে ওয়েইট ভেস্টার, ঠিক যেরকম আমরা লিনিয়ার রিগ্রেশন করার সময় দেখেছিলাম, তাহলে \tilde{W} হবে W -এর অগমেন্টেড রূপ।

এখন, ডিসিশন বাউন্ডারি হাইপার প্লেন বের করার জন্য আগে আমাদের \tilde{W} বের করে নিতে হবে, এর সূত্র হচ্ছে –

$$\tilde{W} = \sum_i \alpha_i S_i$$

এখন, এই সমীকরণে সব মান বসিয়ে পাই,

$$\begin{aligned} \tilde{W} &= (-3.25) \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} + (-3.25) \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix} + (3.5) \begin{pmatrix} 4 \\ 0 \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} 1 \\ 0 \\ -3 \end{pmatrix} \end{aligned}$$

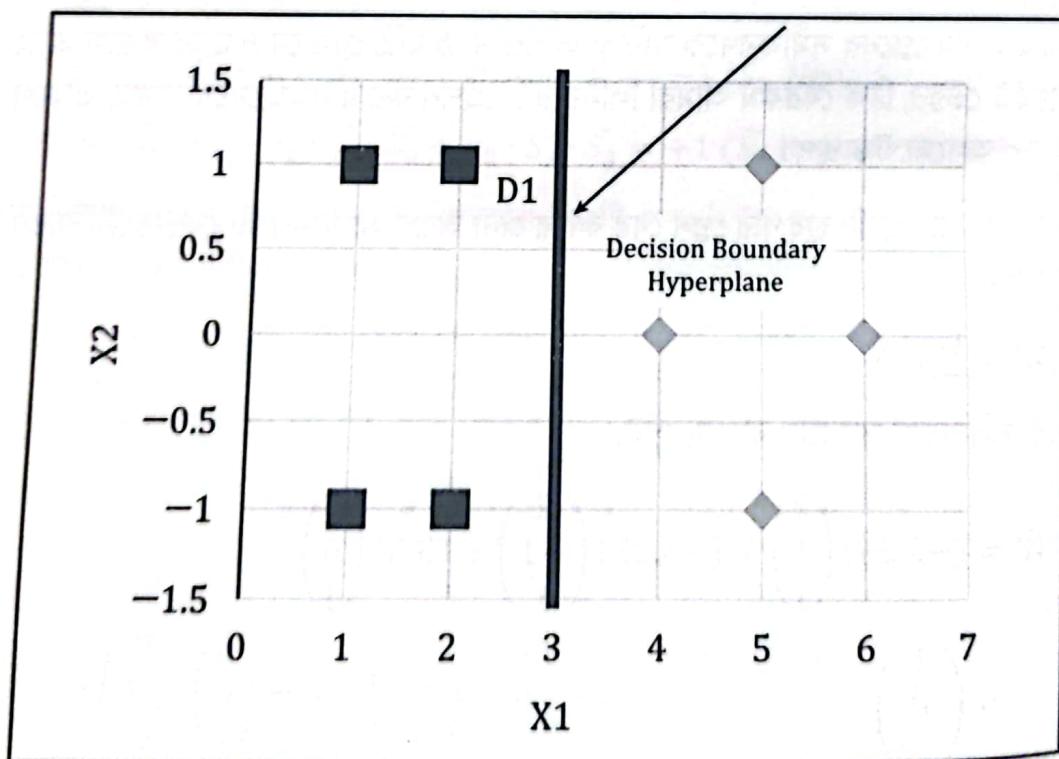
এখন, আমরা বলেছিলাম যে আমরা অগমেন্টে '1' ব্যবহার করছি বায়াস হিসেবে, যেটি একেবারে শেষের রো-এর অতিরিক্ত '1'-টিকে বোঝায়, \tilde{S}_1, \tilde{S}_2 ও \tilde{S}_3 -এর ক্ষেত্রে।

সুতরাং সে হিসেবে ধরতে গেলে, $\tilde{W} = \begin{pmatrix} 1 \\ 0 \\ -3 \end{pmatrix}$ -এ -3 হচ্ছে বায়াস এবং $W = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ ।

দুটো বিষয় একটু মনে রাখতে হবে –

- বায়াসের মান ঝণাত্মক মানে অফসেট হবে ধনাত্মক, আর বায়াসের মান ধনাত্মক না, অফসেট হবে ঝণাত্মক। অফসেট বলতে এখানে সরণ বোঝানো হচ্ছে। অর্থাৎ, ডিসিশন বাউন্ডারি কোনদিকে কতটুকু সরবে, সেটি।
- $W = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ মানে ভার্টিকাল বা উল্লম্বরেখা আর $W = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ মানে হরাইজন্টাল আনুভূমিক রেখা।

এখন এই W এবং বায়াসের মান $y = wx + b$ সমীকরণে বসিয়ে, আমরা পেয়ে যাব আমাদের কার্ডিক্ষন ডিসিশন বাউন্ডারি হাইপারপ্লেন। এখানে লক্ষণীয় বিষয় যে, বায়াস যদি 3 হতো, তাহলে ডিসিশন বাউন্ডারি থাকত X -অক্ষের ঝণাত্মক দিকে, 3 ঘর সরে। আর যদি $W = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ হতো তাহলে ডিসিশন বাউন্ডারি হতো X -অক্ষের সমান্তরাল। এ দুটো শুধুই কিছু শর্টকাট, সহজে অল্প সময়ে ডিসিশন বাউন্ডারি বের করার জন্য। ডিসিশন বাউন্ডারি যদি দুটোর একটিও না হয়, তাহলে আমাদের সাধারণভাবে সমীকরণে বসিয়ে সমাধান করতে হবে।



গ্রাফ 5.2.1

শেষমেশ বাকি থাকে, নতুন পয়েন্টকে ক্লাসিফাই করা।

অধ্যায় ৫ : সাপোর্ট ভেক্টর মেশিন (Support Vector Machine - SVM)

ধরা যাক, নতুন একটি পয়েন্ট $x = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$

$$\text{আমাদের ডিসিশন বাউন্ডারি, } y = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} - 3 = 1 - 3 = -2$$

এখন, যেহেতু, $y \leq 0$ সুতরাং এটি অবশ্যই নেগেটিভ ক্লাস অর্থাৎ Class 1-এর অন্তর্ভুক্ত, যেটি আমরা শুরুতেই বলে নিয়েছি। যদি, $y > 0$ হতো, তবে এটি পজিটিভ ক্লাস অর্থাৎ Class 2-এর অন্তর্ভুক্ত হতো।

তাহলে এই হলো লিনিয়ার সাপোর্ট ভেক্টর মেশিন বা LSVM। আশা করি, সবাই বুঝতে পেরেছেন। এই উদাহরণটিতে একটু ভিন্নভাবে সাপোর্ট ভেক্টর মেশিনের ধারণা বোঝানো হয়েছে। এরপরে আমরা দেখব সাপোর্ট ভেক্টর মেশিন কীভাবে নন-লিনিয়ার ডেটার জন্য কাজ করে।

পরিচেদ ৫.৩ : কার্নাল ট্রিক (Kernel Trick) ও নন-লিনিয়ার এসভিএম (Non-Linear SVM)

আমরা একক্ষণ যে ডেটা নিয়ে কাজ করলাম, সেগুলো ছিল লিনিয়ারলি সেপারেবল (Linearly Separable) ডেটা। সহজ করে বলতে গোলে, এ ধরনের ডেটার বৈশিষ্ট্য হচ্ছে, এদেরকে একটি রেখা টেনে আলাদা করে ফেলা যায়। যেরকম, আমাদের আগের ব্যবহৃত ডেটাসেটটি (গ্রাফ 5.3.1 - মোটা দাগ টেনে দিয়ে আলাদা করে ফেলা হয়েছে দুই ক্লাসের ডেটাকে)।

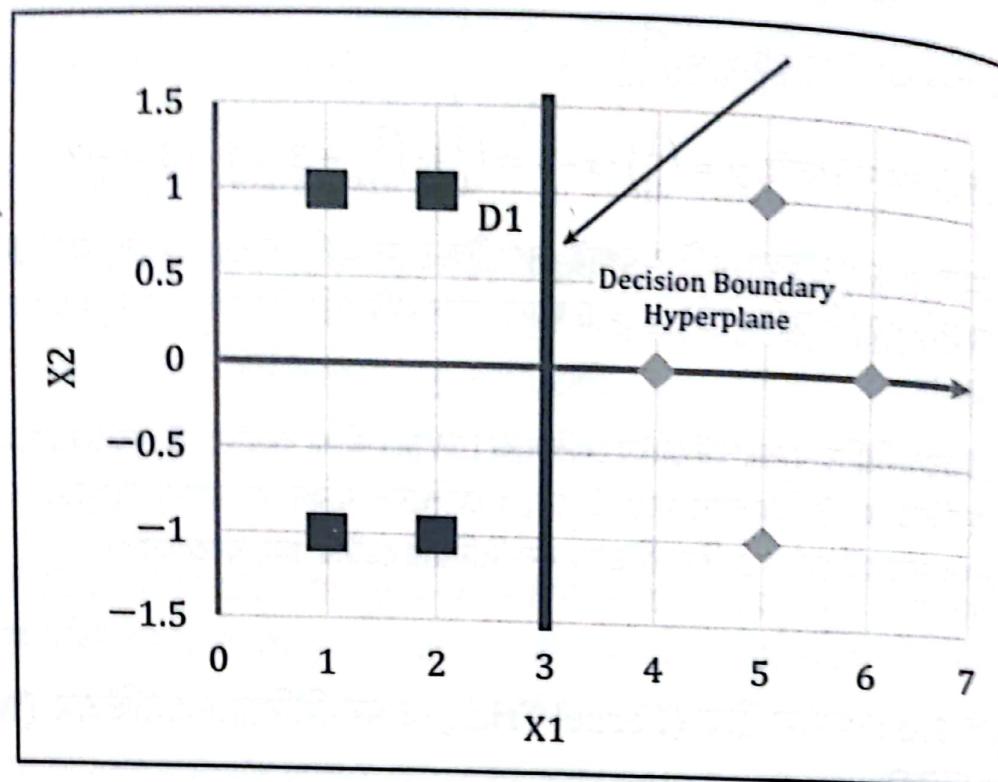
এখন, যদি ডেটাসেট এত সোজা না হয়? যদি, ডেটাসেটের ডেটা একটিমাত্র লিনিয়ার ডিসিশন বাউন্ডারি দিয়ে আলাদা করা না যায়? গ্রাফ 5.3.2' দেখুন।

যদি ডেটাসেট হয় এরকম? তাহলে কীভাবে আলাদা করা যাবে দুটি ক্লাসের ডেটাকে? একটিমাত্র সোজা দাগ টেনে তো এটি সম্ভব নয়। তাহলে?

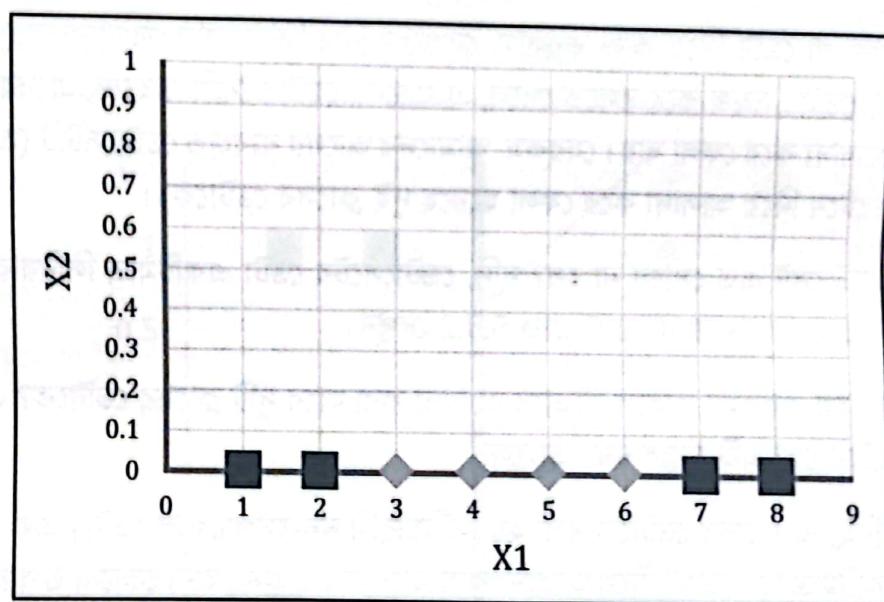
প্রথমে বলে নিই, এ ধরনের ডেটাকে বলা হয় লিনিয়ারলি নন-সেপারেবল ডেটা। এর অর্থ হচ্ছে, এদেরকে একটিমাত্র সরলরেখা দিয়ে আলাদা করা যাবে না। এজন্য, অন্য কোনো উপায় অবলম্বন করতে হবে।

এই অন্য উপায়ের একটি গালভরা নাম আছে - 'কার্নাল ট্রিক' (Kernel Trick)।

কার্নাল ট্রিক ব্যবহার করে, এরকম লিনিয়ারলি নন-সেপারেবল ডেটাকে আমরা SVM-এর সাহায্যে ক্লাসিফাই করতে পারি।



ଆକ୍ଷମିକ ୫.୩.୧



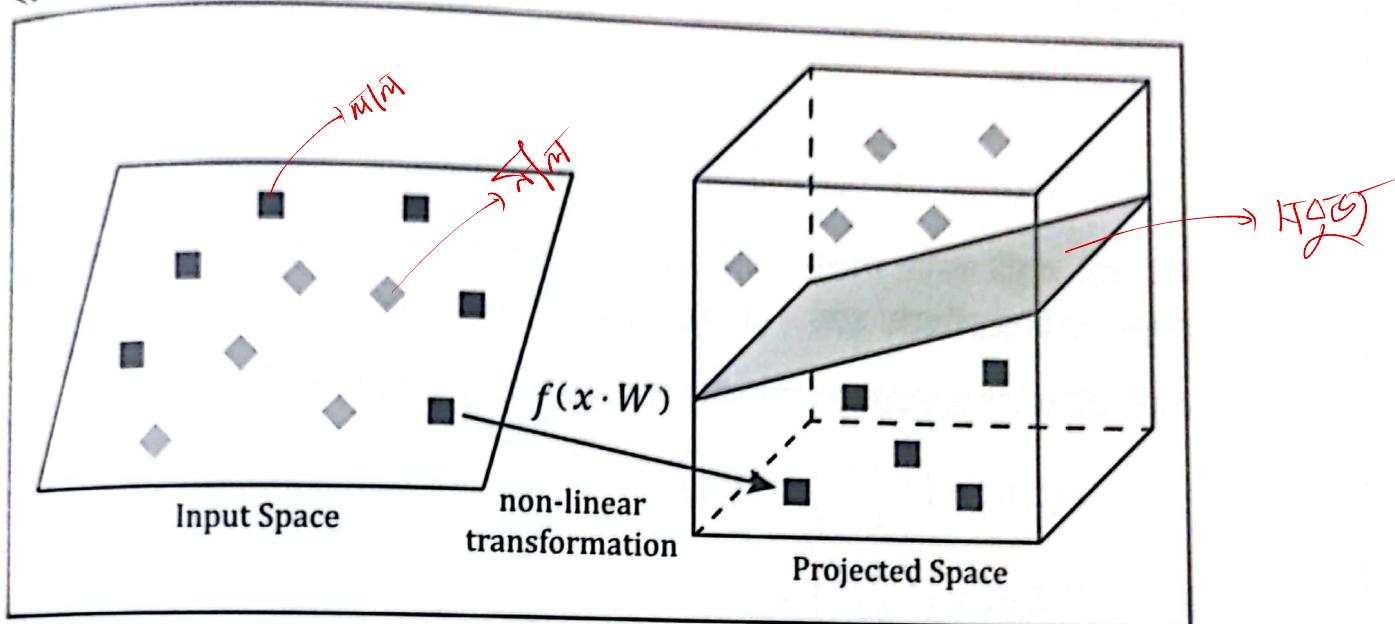
জৰি ৫.৩.২

কার্নাল ট্রিক ব্যাপারটি একটু নয়, বেশ জটিল। এর ভেতরকার থিওরি দেখাতে হলে অনেক গভীর গিয়ে অনেকগুলো গণিতিক জটিল তত্ত্ব বোঝাতে হবে, যেটি আমাদের এই বইয়ের আওতা বাইরে বলা যায়। তাই আমরা অন্তত এই বইয়ের জন্য সেসব জটিল গণিতিক হিসাবনিকাশ করে দিয়ে খুব সহজভাবে আসল বিষয়টি কী ঘটছে সেটি বোঝার চেষ্টা করব। আমাদের মূল লক্ষ্য হবে

অধ্যায় ৫ : সাপোর্ট ভেক্টর মেশিন (Support Vector Machine - SVM)

শুধু বিষয়টি সহজ করে বুনো রাখা, যাতে সেটি আমরা পাইথনের মেশিন লার্নিং লাইব্রেরি ব্যবহার করার সময় বুকাতে পারি যে কী করছি।

যা-ই হোক, প্রথমেই বললাম যে আমাদের ডেটা এবার হচ্ছে লিনিয়ার নন-সেপারেবল ডেটা। এদেরকে আলাদা করার জন্য আমাদেরকে কার্নাল ট্রিক নামে এক ধরনের উপায় অবলম্বন করতে হবে।



ছবি 5.3.3

ছবি 5.3.3 দেখলে ধারণা পাবেন যে, প্রকৃতপক্ষে আমরা কী করতে চাইছি। ওপরে ছবিতে, দেখুন, আমরা যদি ইনপুট স্পেস (Input Space)-এর দিকে তাকাই (যে ডেটা আমরা ইনপুট দিচ্ছি), তাহলে দেখব যে লাল ও নীল এই দুই ক্লাসের বিন্দুকে আমরা শুধু একটি সরলরেখা টেনে আলাদা করতে পারব না। একটু লক্ষ করবেন, আমাদের ইনপুট স্পেস এখানে দ্বিমাত্রিক বা 2D, তাই আমরা একে একমাত্রিক বা 1D সরলরেখা দিয়ে আলাদা করার কথা বলেছি। যদি ত্রিমাত্রিক বা 3D ডেটা হতো, তবে তাদেরকে একটি দ্বিমাত্রিক বা 2D প্লেন এঁকে আলাদা করার কথা বলতাম, যেটি ছবিতে প্রোজেক্টেড স্পেস (Projected Space)-এ দেখানো হয়েছে।

এখন, আমরা এখানে একটি ফাংশন $f()$ ব্যবহার করেছি, যার মধ্যে আমরা আমাদের ইনপুট স্পেসের ডেটা ইনপুট দিয়েছি এবং এই ফাংশনটি সেই 2D ইনপুট ডেটাকে পরিবর্তন করে, ডেটাকে উচ্চতর ডাইমেনশনে নিয়ে গিয়ে 2D ডেটাকে 3D ডেটায় পরিবর্তন করেছে। ডেটাকে 2D থেকে 3D-তে পরিবর্তন করার পরে (মনে করুন, মেরোতে অনেকগুলো বল পড়ে আছে, অর্থাৎ, বলগুলো দ্বিমাত্রিক অবস্থানে আছে; আমরা সবগুলো বল নিয়ে বিভিন্ন বেগে ওপরের দিকে ছুড়ে দিয়েছি, এখন তারা একটি ত্রিমাত্রিক অবস্থানে আছে) দেখা গেল, এখন আমরা, খুব সহজেই একটি 2D প্লেন (সবুজ রঙের) দিয়ে লাল এবং নীল ক্লাসের ডেটাকে আলাদা করতে পারছি। এখানে 2D

থেকে 3D-তে নিয়ে যাওয়ার জন্য যে ম্যাজিক ফাংশনটি আমরা ব্যবহার করলাম, সেটিই হচ্ছে
কার্নাল ফাংশন।

কার্নাল ফাংশন, সহজ করে বললে এক ধরনের ম্যাপিং ফাংশন। সেই ফাংশনের ভেতরে আম
কিছু সংখ্যা ইনপুট দেব, কার্নাল ফাংশন সেই ইনপুট ডেটা নিয়ে কিছু গুণ-ভাগ করে আমাদের
নতুন কতগুলো সংখ্যা ফেরত দেবে।

যেমন ধরা যাক,

$$F(x_1, x_2) = \begin{pmatrix} x_1 + x_2 \\ x_1^2 \end{pmatrix}$$

ধরি, এটি একটি কার্নাল ফাংশন। এটি যেভাবে কাজ করবে, আমাদের নন-লিনিয়ার গ্রাফ থেকে
দেখুন – ডেটাপয়েন্টগুলো হচ্ছে – $(1, 0), (2, 0), (3, 0)$ ইত্যাদি।

এখন, এই ডেটাপয়েন্টগুলোকে যদি আমি এক এক করে কার্নেলে ইনপুট দিই –

$$F(1, 0) = \begin{pmatrix} 1 + 0 \\ 1^2 \end{pmatrix} = (1, 1)$$

$$F(2, 0) = \begin{pmatrix} 2 + 0 \\ 2^2 \end{pmatrix} = (2, 4)$$

$$F(3, 0) = \begin{pmatrix} 3 + 0 \\ 3^2 \end{pmatrix} = (3, 9) \text{ ইত্যাদি।}$$

তাহলে, এইগুলো হবে আমাদের নতুন ফিচার ডেটা পয়েন্ট। এই নতুন ফিচার ডেটা পয়েন্ট নি
আমরা কাজ করব। আমাদের লক্ষ্য হবে এমন একটি কার্নাল ফাংশন খুঁজে বের করা, যেটি ব্যবহ
করে আমরা যে নতুন ফিচার ডেটা পয়েন্টগুলো পাব সেগুলো যদি গ্রাফে প্লট করি তাহলে দেখ
যে আমাদের নতুন ফিচার ডেটা এমনভাবে প্লট হয়েছে, যাতে তাদের আমরা লিনিয়ারলি সেপারে
করতে পারি (একটি সরলরেখা টেনে, কিংবা একটি হাইপার প্লেন দিয়ে)।

সাধারণত কার্নাল ফাংশনে আরেকটি কাজ করা হয়, সেটি হচ্ছে, আমাদের মূল ইনপুট ডেটার
ডাইমেনশন, কার্নাল ফাংশন ব্যবহার করে ফিচার তৈরি করে নেওয়ার সময় সেই ডেটা
ডাইমেনশন বাড়িয়ে দেওয়া হয়।

যেরকম ধরা যাক,

$$F(x_1, x_2) = \begin{pmatrix} x_1^2 \\ x_2^2 \\ x_1^2 + x_2^2 \end{pmatrix}$$

অধ্যায় ৫ : সাপোর্ট ভেস্ট মেশিন (Support Vector Machine - SVM)

এখন দেখুন, আমাদের ইনপুট ডেটার ডাইমেনশন ছিল 2, কিন্তু নতুন পরিবর্তিত ডেটার ডাইমেনশন 3; ঠিক যেরকম আমরা শুরুর উদাহরণে দেখলাম। এই নতুন হায়ার-ডাইমেনশনাল ডেটাই আমাদের এখন ফিচার হিসেবে ব্যবহৃত হবে।

এখন প্রশ্ন হচ্ছে, আমরা কীভাবে বুঝব কোন ফাংশন কার্নাল হিসেবে ব্যবহার করতে হবে?
সাধারণত, সাপোর্ট ভেস্ট মেশিনের ক্ষেত্রে কিছু জনপ্রিয় কার্নাল আছে, সেগুলোই সাধারণ ক্ষেত্রে ঘুরেফিরে ব্যবহার করা হয়। আর ক্ষেত্রবিশেষে, নতুন আরো অপটিমাইজড কোনো কার্নাল তৈরি করে নেওয়া হয় ডেটাসেটের ধরনের ওপরে ভিত্তি করে।

আমরা প্রথমেই একটি কার্নাল ফাংশন নিয়ে কাজ করব, একে বলা হয় গাউসিয়ান কার্নাল (Gaussian Kernel)। এই কার্নাল ব্যবহার করে, প্রতিটি ফিচার ডেটা বের করার সূত্র হচ্ছে –

$$f_i = e^{-\frac{\|X-X_i\|}{2\sigma^2}}$$

x_1	x_2
2	3
3	4
4	5

টেবিল 5.5.1

ধরা যাক, টেবিল 5.5.1 আমাদের ডেটাসেট। এখন, আমরা এই ডেটাসেটের (2, 3) পয়েন্টটির জন্য গাউসিয়ান কার্নাল ব্যবহার করে নতুন ফিচার পয়েন্ট বের করব। আমাদের এই পয়েন্টের ইনপুট ডেটাতে ফিচার আছে দুটি, $x_1 = 2$ ও $x_2 = 3$ ।

এখন, আমরা যদি এই ডেটা পয়েন্টকে আমাদের গাউসিয়ান কার্নালে ইনপুট হিসেবে দিই, তাহলে আমাদের নতুন 3টি ফিচার তৈরি হবে, যেহেতু আমাদের ট্রেনিং ডেটার সংখ্যা 3টি। নতুন ফিচারগুলো হলো –

$$f_1 = e^{-\frac{\|X-X_1\|}{2\sigma^2}} = e^{-\frac{(2-2)^2+(3-3)^2}{2\sigma^2}} \approx 1$$

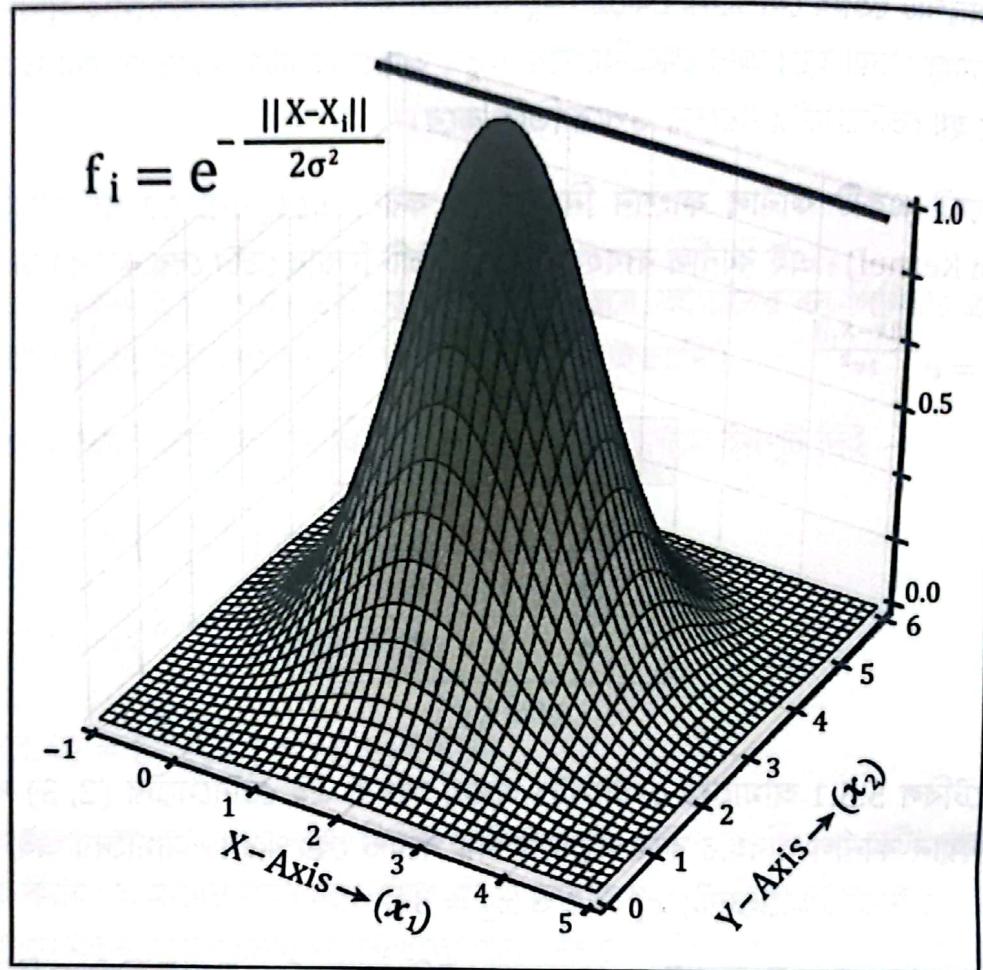
$$f_2 = e^{-\frac{\|X-X_2\|}{2\sigma^2}} = e^{-\frac{(2-3)^2+(3-4)^2}{2\sigma^2}} \approx 0$$

$$f_3 = e^{-\frac{\|X-X_3\|}{2\sigma^2}} = e^{-\frac{(2-4)^2+(3-5)^2}{2\sigma^2}} \approx 0$$

এই হিসাবগুলোতে σ^2 -এর একটি মান আমরা ধরে নিয়েছি 1। সূতরাং, আমরা নতুন তিনটি ফিচার ভ্যালু পেয়ে গেলাম,

$$(f_1, f_2, f_3) \approx (1, 0, 0)$$

এরপর থেকে $(2, 3)$ পয়েন্টটির বদলে এর নতুন ফিচার ভেস্টের $f = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$ নিয়েই আমাদের করতে হবে। নিচের ছবিটি দেখি (ছবি 5.3.4) :



ছবি 5.3.4

গ্রাফটিতে দেখা যাচ্ছে, ফিচার-এর মান '1' কেবল তখনই হয়, যখন আমরা X-অক্ষ ও Y-অক্ষের যথাক্রমে 2 ও 3 একক নিছি। বাকি ডেটা পয়েন্ট দুটির ক্ষেত্রে ফিচারের মান শূন্য কাছাকাছি।

এ থেকে বোঝা যাচ্ছে, আমাদের এই কার্নালটি আসলে যা করছে তা হলো, এটি যে পয়েন্ট ইন্পুট হিসেবে নিচ্ছে, তার সঙ্গে ট্রেনিং ডেটার সব পয়েন্টের সামঞ্জস্য (Similarity) বের করছে। যখন ট্রেনিং ডেটার মধ্যেকার পয়েন্ট $(2,3)$ -এর সঙ্গে, অর্থাৎ নিজের সঙ্গে সামঞ্জস্য খুঁজে বের করা তখনই কেবল ফিচারের মান '1' হচ্ছে, অন্যান্য ক্ষেত্রে শূন্যের কাছাকাছি হচ্ছে।

এখন ধরে নিই, আমাদের হাইপোথিসিস আগের মতোই –

অধ্যায় ৫ : সাপোর্ট ভেক্টর মেশিন (Support Vector Machine - SVM)

$$h_w(x) = \begin{cases} 0, & W^T x < 0 \\ 1, & W^T x \geq 0 \end{cases}$$

গুরু যে পরিবর্তন আসবে, তা হলো x (ইনপুট-এর সময় যা ফিচার হিসেবে ধরেছিলাম)-এর বদলে আমরা f (নতুন ফিচার ভেক্টর) নেব।

তাহলে, আমাদের নতুন হাইপোথিসিস দাঁড়াবে –

$$h_w(x) = \begin{cases} 0, & W^T f < 0 \\ 1, & W^T f \geq 0 \end{cases}$$

বাকি থাকল, আমরা W অর্থাৎ আমাদের প্যারামিটার সেট কীভাবে পাব এবং আমাদের অপটিমাইজেশন ফাংশন কী হবে? প্যারামিটার সেট পাব ঠিক আগের নিয়মে –

প্রথম প্যারামিটার w_0 -এর জন্য,

$$w_0 = w_0 - \alpha \frac{\partial}{\partial w_0} J(W)$$

$$\frac{\partial}{\partial w_0} J(W) = \frac{1}{m} \sum_{l=1}^m \{h_w(x^{(l)}) - y^{(l)}\} \cdot x_0^{(l)}$$

প্রথম প্যারামিটার ($j = 1, 2, 3, \dots n$)-এর জন্য,

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(W)$$

$$\frac{\partial}{\partial w_j} J(W) = \frac{1}{m} \sum_{l=1}^m \{h_w(x^{(l)}) - y^{(l)}\} \cdot x_j^{(l)} - \frac{\lambda}{m} w_j$$

আর অপটিমাইজেশন ফাংশন আমাদের LSVM-এর ক্ষেত্রে ছিল –

$$\min_w C \left[\sum_{l=1}^m \{y^{(l)} (\text{Cost}_1(z)) + (1 - y^{(l)}) (\text{Cost}_0(z))\} \right] + \frac{1}{2} \sum_{j=1}^n w_j^2$$

যেখানে, $z = W^T x$ । এখন, x -এর বদলে শুধু f ব্যবহার করব, যেহেতু এটি আমাদের নতুন ফিচার ভেক্টর। সুতরাং, নন-লিনিয়ার সাপোর্ট ভেক্টর মেশিনের জন্য আমাদের নতুন অপটিমাইজেশন ফাংশন দাঁড়ায় –

$$\min_w C \left[\sum_{l=1}^m \left\{ y^{(l)} (\text{Cost}_1(W^T f^{(l)})) + (1 - y^{(l)}) (\text{Cost}_0(W^T f^{(l)})) \right\} \right] + \frac{1}{2} \sum_{j=1}^n w_j^2$$

নবশেষ্যে আরো কয়েকটি বিষয় বিবেচনা করতে হবে :

মেশিন লার্নিং অ্যালগরিদম

1. $C (= \frac{1}{\lambda})$ -এর মান নেওয়ার ক্ষেত্রে, যদি C -এর মান অনেক বড়ো নিই, তাহলে আমরা High Bias, High Variance দেখতে পাব, অর্থাৎ ওভারফিটিং হওয়ার সম্ভাবনা থাকবে বেশি। যদি C -এর মান অনেক ছোটো নিই, তাহলে আমরা High Bias, Low Variance দেখতে পাব, অর্থাৎ আন্ডারফিটিং হওয়ার সম্ভাবনা থাকবে বেশি।
2. σ^2 -এর মান অনেক বড়ো হলে, High Bias, Low Variance, আর মান অনেক ছোটো হলে Low Bias, High Variance। এর কারণ হচ্ছে, σ^2 -এর মান অনেক বড়ো হলে ফিচার Smoothly Vary করতে পারবে, আর ছোটো হলে তা পারবে না।
3. গাউসিয়ান কার্নালের মতো আরো কিছু জনপ্রিয় কার্নাল আছে –
 - Polynomial Kernel:

$$K(x_1, x_2) = (x_1 \cdot x_2 + 1)^d \quad [d = \text{degree of polynomial}]$$
 - Gaussian Radial Basis Function (RBF):

$$K(x_1, x_2) = e^{-\gamma |x_1 - x_2|^2} \quad [\text{for } \gamma > 0, \text{ mostly } \gamma = \frac{1}{2\sigma^2}]$$
 - Laplace RBF Kernel:

$$K(x_1, x_2) = e^{-\frac{|x_1 - x_2|}{\sigma}}$$
 - Linear Kernel (No Kernel at all):

$$h_w(x) = \begin{cases} 0, & W^T x < 0 \\ 1, & W^T x \geq 0 \end{cases}$$

এই ছিল আমাদের সাপোর্ট ভেক্টর-সম্পর্কিত যাবতীয় আলোচনা।