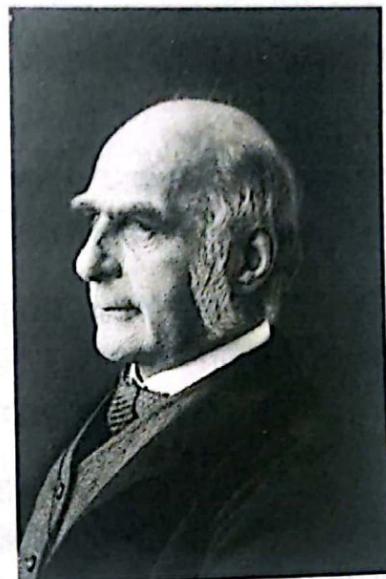


অধ্যায় ৩ : লিনিয়ার রিগ্রেশন (Linear Regression)

লিনিয়ার রিগ্রেশন শব্দের সঙ্গে অনেকে হয়তো পরিচিত, অনেকে নয়। যাঁরা পরিসংখ্যান বা স্ট্যাটিস্টিকস (Statistics) পড়েছেন, তাঁরা হয়তো জানেন এই পদ্ধতিটি সম্পর্কে। এটি আসলে পরিসংখ্যানের একটি পদ্ধতি। আর বর্তমানে এটি মেশিন লার্নিং অ্যালগরিদমগুলোর মধ্যে অন্যতম জনপ্রিয় একটি পদ্ধতি। এটি এত জনপ্রিয় হওয়ার মূল কারণ হয়তো এর সহজবোধ্যতা।

রিশ্বনের উভাবক বলা হয় স্যার ফ্রান্সিস গ্যালটনকে (Francis Galton, 1822 - 1911)। তিনিই প্রথম পরিসংখ্যানে এই রিশ্বনের ব্যবহার করেন।



ଭବି ୩.୧ : Francis Galton (1822 – 1911)

পরিচ্ছদ ৩.১ : লিনিয়ার রিপ্রেশন কী

তাত্ত্বিক আলোচনায় যাওয়ার আগে, আমাদের দৈনন্দিন জীবনের সঙ্গে মিলে যায়, এরকম একটি ঘটনা দিয়ে বোঝার চেষ্টা করা যাক লিনিয়ার রিপ্রেশন ব্যাপারটি কী। এটি আসলে আমার খুব প্রিয় একটি উদাহরণ। ক্লাসে আমার ছাত্রাত্মাদের বোঝানোর সময়েও আমি এই উদাহরণটি ব্যবহার করি। এই উদাহরণটি আগেও একবার বর্ণনা করা হয়েছে। আপনাদের সুবিধার্থে আমি আবারও এটি বর্ণনা করছি এখানে।

এটি বর্ণনা করছি এখানে।
ধরা যাক, আপনাকে বিভিন্ন সাইজের (ইঞ্জিন) পিংজার দাম দেওয়া আছে (আগের অধ্যায়ে দেওয়া উদাহরণের মতো)। এই ডেটার ওপরে ভিত্তি করে আপনাকে এখন কাজ করতে হবে। আপনার কাছে থাকা দেউ ট্রেবিল 3.1.1-এ দেওয়া হলো।

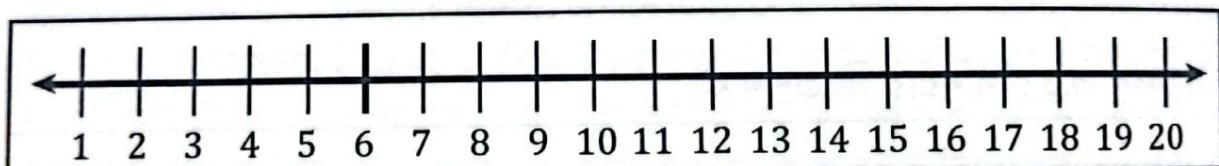
কাছে থাকা ডেটা টেবিল 3.1.1-এ দেওয়া হলো।
এখন এই দামগুলোর ওপর ভিত্তি করে, আপনাকে যদি আন্দাজ করতে দেওয়া হয় যে 17 ইঞ্চি
কোনো একটি পিংজা যদি বানানো হয়, তবে সেই পিংজার দাম কত হবে, তাহলে কি আপনি সেটি
আন্দাজ করতে পারবেন? যদি পারেন তাহলে সেটি কীভাবে করবেন?

পিংজার সাইজ (ইঞ্চিতে)	পিংজার দাম (টাকায়)
6"	350/-
8"	775/-
12"	1150/-
14"	1395/-
18"	1675/-

টেবিল 3.1.1

আপনি প্রথমেই দেখবেন যে 17 সংখ্যাটির বৈশিষ্ট্য কী। এটি 14-এর চেয়ে বড়ো, কিন্তু 18-এর চেয়ে ছোটো, তাই না? তার মানে 17 ইঞ্চি পিংজার দাম অবশ্যই 14 ইঞ্চি পিংজার দামের চেয়ে বেশি হবে, কিন্তু 18 ইঞ্চি পিংজার দামের চেয়ে কম হবে।

আবার দেখুন, আমরা যদি একটি সংখ্যারেখার কথা চিন্তা করি (ছবি 3.1.1), তাহলে আমরা খুব সহজেই এটি বলতে পারি যে 17 সংখ্যাটি 14 ও 18-এর মধ্যে 18-এর বেশি কাছাকাছি, তাই না? এই তথ্যের ওপর ভিত্তি করে আমরা এও বলতে পারি, এখন যে 17 ইঞ্চি পিংজার দাম 18 ইঞ্চি পিংজার দামের কাছাকাছি কিছু একটা হবে।



ছবি 3.1.1

একটু ভালো করে চিন্তা করে দেখুন, আপনাকে কেউ যদি 17 ইঞ্চি পিংজার দাম আন্দাজ করতে দিত, আপনিও মনে মনে ঠিক এভাবেই চিন্তা করতেন। শুধু আপনি অনেক দ্রুত এবং নিজের অজান্তেই এটি চিন্তা করে ফেলেন দেখে ঠিক অনেক সময় বুঝতে পারেন না যে, আসলে চিন্তাটি কীভাবে করলেন।

আপনি যদি ওপরের ঘটনাটি বুঝে থাকেন, তাহলে আপনি লিনিয়ার রিপ্রেশন বুঝে গেছেন অনেকখানিই।

তাহলে লিনিয়ার রিপ্রেশন বিষয়টি আসলে কী? গণিতের ভাষায় বললে এটি অধীন চলক (Dependent Variable) ও স্বাধীন চলকের (Independent Variable) মধ্যেকার সম্পর্ক নির্ণয়ের একটি মাধ্যম।

অধ্যায় ৩ : লিনিয়ার রিগ্রেশন (Linear Regression)

একেবারেই দুর্বোধ্য মনে হচ্ছে, তাই না? আচ্ছা চলুন দেখি, এখন গণিতের ভাষার সঙ্গে আমাদের পিংজার ব্যাপারটি মেলানো যায় কি না।

আপনারা যদি একটু ভালো করে পিংজার সাইজ ও দামগুলো লক্ষ করেন, তাহলে দেখবেন, যে পিংজার সাইজ যত বেড়েছে, তার দাম তত বেড়েছে। তার মানে, পিংজার দাম কত হবে সেটি নির্ভর করছে তার সাইজের ওপর। সাইজ যত বড়ো, দাম তত বেশি। তাহলে, এখানে পিংজার দামটা পিংজার সাইজের ওপর নির্ভরশীল, তাই না?

তাহলে কি আমরা বলতে পারি না যে, এখানে পিংজার সাইজ হলো আমাদের স্বাধীন চলক (Independent Variable) এবং পিংজার দাম হলো অধীন চলক (Dependent Variable)?

আচ্ছা, এই IV (Independent Variable)-কে এখন থেকে IV লিখব, আর Dependent Variable-কে লিখব DV) আর DV-এর মধ্যে কী সম্পর্ক সেটি নিয়ে একটু কথা বলি! এখানে IV আর DV-এর মধ্যে সম্পর্ক বলতে আসলে বোঝাবে IV-এর দাম এক একক (1 unit) বৃদ্ধি পেলে DV-এর দাম কতটুকু বৃদ্ধি পাবে, সেটি নির্ণয়ের চেষ্টা করা! এটিই লিনিয়ার রিগ্রেশন।

এ তো বললাম কঠিন গণিতের ভাষার, এখন খাবারের ভাষায় বলি। ওপরের উদাহরণ থেকে দেখুন, আমাদের পিংজার সাইজ প্রতিবার বিভিন্ন আকারে বৃদ্ধি পেয়েছে। প্রথমে ছিল 6 ইঞ্চি, এর পরে 8 ইঞ্চি, 12 ইঞ্চি, 14 ইঞ্চি, 18 ইঞ্চি ইত্যাদি। আমাদের এখান থেকে একটু বের করার চেষ্টা করতে হবে যে পিংজার সাইজ 1 ইঞ্চি বৃদ্ধি পেলে আসলে দাম গড়ে কতটা বাড়ে। এই হিসাবটি আমরা যত সূক্ষ্মভাবে বের করতে পারব, তত আমাদের হিসাব নির্ভুল হবে।

তাহলে মোদা কথাটা কী দাঁড়াল? স্বাধীন চলকের মানের পরিবর্তনের সঙ্গে সঙ্গে কীভাবে অধীন চলকের মানের পরিবর্তন হয়, অর্থাৎ স্বাধীন ও অধীন চলকের মধ্যেকার সম্পর্কটি আসলে কী, সেটি নির্ণয়ের একটি পদ্ধতিই হলো লিনিয়ার রিগ্রেশন! এবং এই সম্পর্কটি যদি আমরা একবার বের করে ফেলতে পারি আমাদের জানা অধীন ও স্বাধীন চলকের মান থেকে, তাহলে সেই সম্পর্ক ব্যবহার করে আমরা স্বাধীন চলকের যে-কোনো অজানা মানের জন্য অধীন চলকের মান বের করে ফেলতে পারি।

অর্থাৎ, এই যে 5টি ভিন্ন ভিন্ন সাইজের পিংজার দাম ও সাইজ যে আমাদের দেওয়া ছিল, সেখান থেকে আমরা যদি সাইজ ও দামের মধ্যে একটি সহজ-সরল সম্পর্ক বের করে ফেলতে পারি, তাহলে সেই সম্পর্ক ব্যবহার করে খুব সহজেই এমন কোনো সাইজের পিংজা, যার দাম আমরা জানি না (যেমন 17 ইঞ্চি পিংজা), তার দামও আন্দাজ করে বলে দিতে পারি। দামটি পুরোপুরি সঠিক না হলেও, খুব কাছাকাছি একটি মান হবে। আমাদের এখনকার কাজের জন্য সেটুকুই যথেষ্ট।

পরিচ্ছেদ ৩.২ : নিউমেরিকাল অ্যানালাইসিস (Numerical Analysis) ব্যবহার করে লিনিয়ার রিগ্রেশন

লিনিয়ার রিগ্রেশন দুই পদ্ধতিতে করা যায়। প্রথম পদ্ধতি অপেক্ষাকৃত সহজ, শুধু সূত্র প্রয়োগ করতে হবে, আর কিছু নয়। এটি একটি নিউমেরিকাল অ্যানালাইসিস মেথড। তাই এটি দিয়েই শুরু করি।

আমাদের টেবিল 3.1-এ দেওয়া ডেটাসেটটি নিয়েই আমরা কাজ করব এবং এটি দিয়েই আমাদের ধারণাটি বুঝব। দেখি কতদূর এগোতে পারি। ব্যবহারের সুবিধার জন্য আমাদের ডেটাসেটটি (পিংজার দাম ও সাইজের টেবিলটিকে আমরা ডেটাসেট বলব) আবার এখানে দিচ্ছি (টেবিল 3.2.1)।

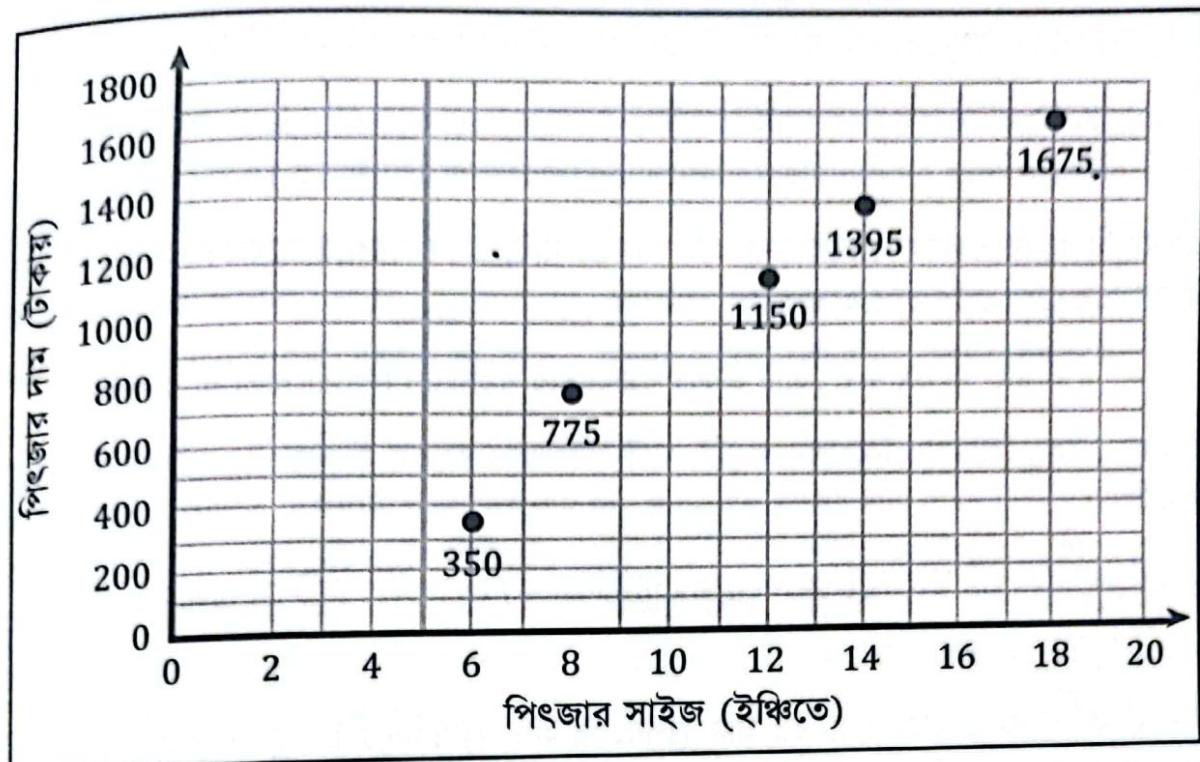
পিংজার সাইজ (ইঞ্চিতে)	পিংজার দাম (টাকায়)
6"	350/-
8"	775/-
12"	1150/-
14"	1395/-
18"	1675/-

টেবিল : 3.2.1

আমরা বলেছিলাম, আমাদের লক্ষ্য হচ্ছে, যে ডেটাসেট আমাদের কাছে আছে, সেটি ব্যবহার করে আমাদের বের করতে হবে 17 ইঞ্চি পিংজার দাম কত টাকা, তাই না?

দাম বের করার আগে আমরা যদি একটি গ্রাফ আঁকার চেষ্টা করি, যেখানে X-অক্ষ বরাবর নেব আমাদের স্বাধীন চলক, যেটি হলো পিংজার সাইজ এবং Y-অক্ষ বরাবর নেব আমাদের অধীন চলক, যা হলো পিংজার দাম। তাহলে আমাদের গ্রাফের চেহারাটি দাঁড়াবে ছবি 3.2.1-এর গ্রাফের মতো।

আশা করছি, গ্রাফটি সবাই বুঝতে পেরেছেন? এই গ্রাফে দেখুন 5টি বিন্দু বা পয়েন্ট আছে, আমরা এগুলোকে বলব আমাদের ডেটা পয়েন্ট। এখন আসি মূল কথায়। আমাদের যেটি করতে হবে, এই ডেটাপয়েন্টগুলোর মধ্যে আমাদের একটি সম্পর্ক বের করে ফেলতে হবে, তাই না? যদি ঠিক বের করতে পারি, তাহলেই কেবল সেই সম্পর্কটি ব্যবহার করে আমরা নতুন অজানা সাইজের পিংজার দাম বের করে ফেলতে পারব।



গ্রাফ 3.2.1: পিংজার সাইজ ও দামের মধ্যে সম্পর্ক

এখন, কতগুলো বিন্দুর মধ্যে সবচেয়ে সহজ-সরল সম্পর্ক কী হতে পারে? উভয় হচ্ছে – একটি সরলরেখা। আমরা যদি এমন একটি সরলরেখা বের করতে পারি, যার ওপরে আমাদের সবগুলো ডেটা পয়েন্ট অবস্থান করে, তাহলে কী হবে? ওই সরলরেখার সমীকরণে প্রতিটি বিন্দুর মান বসালে সমীকরণের দুই পক্ষ সমান পাব। মনে করে দেখুন, ছোটোবেলায় আমরা পড়েছিলাম, যে-কোনো সরলরেখার সমীকরণে ওই সরলরেখার ওপরের কোনো বিন্দুর মান বসালে আমরা সমীকরণটিকে সিদ্ধ করতে পারি, এরকম একটি কথা ছিল।

তাই, আমাদেরকে একটি সরলরেখা আঁকার চেষ্টা করতে হবে। লক্ষ্য থাকবে এমন একটি সরলরেখা খুঁজে বের করা, যা আমাদের সবগুলো ডেটা পয়েন্টের ওপর দিয়ে যায়, অর্থাৎ সবগুলো ডেটা পয়েন্ট আমাদের সরলরেখাটির সমীকরণকে সিদ্ধ করে। সেরকম যদি কোনো সরলরেখা না-ডেটা পয়েন্টের দূরত্ব সর্বনিম্ন হয়।

এখানে লক্ষণীয়, আমরা হয়তো-বা সরলরেখা না এঁকে আঁকাবাঁকা বক্ররেখা বা কার্ড (Curve) আঁকার চেষ্টা করতে পারি। কেননা এমনও হতে পারে যে, আমরা একটি সরলরেখা আঁকলে সেটি সবগুলো ডেটা পয়েন্ট দিয়ে যাচ্ছে না কিন্তু যদি একটি কার্ড আঁকি, তাহলে সেটি সবগুলো ডেটা পয়েন্ট দিয়ে যাচ্ছে। কিন্তু, সে ক্ষেত্রে তাহলে সেটি আর লিনিয়ার বা সরলরেখিক থাকবে না (সমীকরণের হিসেবে যদি বলি, x -এর সর্বোচ্চ ঘাত আর ১ থাকবে না, এর বেশি হয়ে যাবে)।

একটু মনে করে দেখুন, ছোটোবেলায় আমাদের পড়া সরলরেখার সমীকরণ ছিল,

$$Y = mX + c \quad (\text{এখানে } X \text{ হচ্ছে স্বাধীন চলক, } Y \text{ হচ্ছে } X\text{-এর ফাংশন এবং সমীকরণটি লিনিয়ার, কেননা স্বাধীন চলকের সর্বোচ্চ ঘাত 1})$$

এখন যদি আমরা কোনো বক্ররেখা বা কার্ড আঁকতে যাই, তখন অবশ্যই আমাদের X -এর ঘাতের মান 1-এর চেয়ে বেশি হতে হবে, যেমন,

$$Y = aX^2 + bX + C, \text{ এটি একটি বক্ররেখার সমীকরণ (প্যারাবোলা বা অধিবৃত্ত)}$$

আমরা যেহেতু লিনিয়ার রিষ্ট্রেশন পড়ছি, তাই আপাতত আমরা সরলরেখার মধ্যেই সীমাবদ্ধ থাকছি।

এখন, আমরা একটি ফাংশন চিন্তা করি, যার নাম দিলাম $f(x)$ । ফাংশনটি দেখতে হবে এরকম –

$$f(X) = Y$$

অর্থাৎ, ফাংশনটিতে আমরা X -এর মান (পিংজার সাইজ) ইনপুট দিলে, ফাংশনটি আমাদের আউটপুট দেবে Y , যেটি কিনা পিংজার দাম। আমরা তো এই জাদুর ফাংশনটিই খুঁজে বের করতে চাইছি, তাই নয় কি? মনে করে দেখুন, আমরা পড়েছিলাম, ফাংশন হচ্ছে এক ধরনের অব্যয় বা সম্পর্ক, আমাদের ইনপুট ও আউটপুটের ভেতরে। আমরা এই সম্পর্কটিই খুঁজে বের করতে চাইছি।

আমরা, $f(X) = mX + c$ লিখতে পারি অবশ্যই, যেহেতু আমরা সরলরেখা বের করতে চাইছি, তাই এটাই হবে আমাদের ফাংশনের গাণিতিক রূপ। তাহলে, শেষমেশ দাঁড়াল –

$$Y = mX + c$$

এখন, আমাদের এই সমীকরণে m হলো ঢাল বা স্লোপ, যেটি সরলরেখার দিক নির্ধারণ করে। মানে, X -অক্ষের ধনাত্মক দিকের সঙ্গে কত ডিগি কোণ করে আছে তার Tangent-এর মানকে বোঝায়। আর c হলো Y -অক্ষের কর্তিত অংশ বা ইন্টারসেক্ষন (Intercept)-এর মান, অর্থাৎ সরলরেখাটি Y -অক্ষকে কোন বিন্দুতে ছেদ করেছে তার মান।

এখন, m ও c -এর ভিন্ন ভিন্ন অসীমসংখ্যক মানের জন্য আমরা কিন্তু অসংখ্য সরলরেখা পাব। সেই সরলরেখাগুলো X -অক্ষের ধনাত্মক দিকের সঙ্গে বিভিন্ন কোণ করে থাকবে এবং তারা Y -অক্ষকে বিভিন্ন বিন্দুতে ছেদ করবে।

আমরা ধরে নিই যে, M_{final} ও C_{final} হচ্ছে m ও $21c$ -এর সেই দুটি বিশেষ মান, যার জন্য আমরা আমাদের কাজিক্ষত সরলরেখাটি পেয়ে যাব যেটি আমাদের সমস্ত ডেটা পয়েন্টের ওপর দিয়ে যাবে। তাহলে, মানগুলো সমীকরণে বসিয়ে পাই –

অধ্যায় ৩ : লিনিয়ার রিফেশন (Linear Regression)

$$Y = (X \times M_{final}) + C_{final}$$

এ ফাংশনটি বের করাই হচ্ছে আমাদের লক্ষ্য বা টার্গেট, তাই আমরা একে বলব টার্গেট (Target) ফাংশন বা ট্রু (True) ফাংশন। এই ফাংশনের সরলরেখাটি আমাদের সব ডেটা পয়েন্টের ওপর দিয়ে যাবে।

কিন্তু কথা হচ্ছে, আমরা তো M_{final} ও C_{final} -এর মান জানি না। তাই, আমরা র্যানডমলি দুটি মান দিয়ে শুরু করব। ধরি সেগুলো m_1 ও c_1 , সুতরাং সরলরেখাটির সমীকরণ তখন হবে,

$$Y = X \times m_1 + C_1$$

হয়তো আমরা শুরুতে দেখলাম, আমাদের এই m_1 ও c_1 -এর মানের জন্য এমন একটি সরলরেখা পেয়েছি, যেটি আমাদের ৫টি ডেটা পয়েন্টের মধ্যে মাত্র একটি ডেটা পয়েন্টের ওপর দিয়ে গেছে, বাকি ৪টির ধারেকাছে দিয়েও যায়নি। তখন আমরা যোগ, বিয়োগ, গুণ, ভাগ ইত্যাদি নানা গাণিতিক অপারেশন চালিয়ে চেষ্টা করব আমাদের m_1 ও c_1 -এর মানগুলোকে M_{final} ও C_{final} -এর মানের যত কাছাকাছি পারা যায় নিয়ে আসতে, যাতে করে আমাদের সরলরেখাটি সবগুলো ডেটা পয়েন্টের ওপর দিয়ে যায়।

এই যে, প্রতিবার আমরা একেকটি m ও c -এর মান পাচ্ছি (যেমন m_1 , c_1 পেলাম ইত্যাদি) সেগুলোকে আমাদের ফাংশনে বসালে আমরা যে সমীকরণটি পাই, তাকে বলা হয় হাইপোথিসিস। সমস্ত সম্ভাব্য হাইপোথিসিস নিয়ে তৈরি হয় হাইপোথিসিস স্পেস (Hypothesis Space)। তার মধ্যে থেকে যেটি সবচেয়ে ভালো ফলাফল দেয়, আমরা তাকেই আমাদের ফাইনাল হাইপোথিসিস হিসেবে বেছে নিই।

যেমন, $Y = X \times m_1 + c_1$, এটি একটি হাইপোথিসিস। এই হাইপোথিসিস হচ্ছে এমন একটি ফাংশন, যেটিকে আমরা আমাদের টার্গেট ফাংশনের সঙ্গে মিলে যেতে পারে বলে ধরে নিই এবং পরে হিসাবনিকাশ করে দেখি যে আসলেই মিলে যাচ্ছে কি না।

আমরা m_1 ও c_1 -এর ওপরে গাণিতিক বিভিন্ন অপারেশন চালিয়ে, ধরি, নতুন আরো দুটি মান পেলাম - m_2 ও c_2 । সুতরাং, তখন আমাদের বিবেচনার জন্য নতুন হাইপোথিসিস হবে

$$Y = X \times m_2 + c_2$$

এবং আমরা আবার মিলিয়ে দেখব সেটি আমাদের টার্গেট ফাংশনের মতো আচরণ করে কি না, অর্থাৎ সবগুলো ডেটাপয়েন্টের ওপর দিয়ে যায় কি না, বা কাছাকাছি যায় কি না।

এই হাইপোথিসিসের ধারণাটি মেশিন লার্নিংয়ে অত্যন্ত গুরুত্বপূর্ণ। তাই আমি বলব, কেউ যদি এখনো এই ধারণাটি পরিষ্কারভাবে না বুঝে থাকেন, তাহলে অনুচ্ছেদের শুরু থেকে আরেকবার পড়ুন।

মেশিন লার্নিং অ্যালগরিদম

এখন, আমরা ইতিমধ্যেই বলেছি, এত এত অসংখ্য সরলরেখার ভিত্তে আমাদের এমন একটি সরলরেখা খুঁজে বের করতে হবে, যে সরলরেখা থেকে **এই ৫টি বিন্দুর দূরত্ব** অন্য যে-কোনো সরলরেখার চেয়ে কম হয়। সেই **সরলরেখাটিকে** আমরা অনেক সময় বলে থাকি **রিগ্রেশন (Regression) লাইন**, এটিই আমাদের কাঞ্জিক্ষত সমাধান।

এই সরলরেখাটির আরো একটি সুন্দর নাম আছে, এটাকে বলা হবে **বেস্ট ফিট লাইন (Best Fit Line)**। এর কারণ হচ্ছে সেই সরলরেখাটা আমাদের ডেটার ভেতরে সবচেয়ে ভালো ফিট করবে কিংবা সবচেয়ে ভালো সমাধান দেবে।

যা-ই হোক, আমরা আমাদের সরলরেখার অনুসন্ধানে ফেরত আসি। সরলরেখার সমীকরণের এই **m** ও **c -এর মান** ডেটাসেট থেকে সরাসরি পাওয়ার জন্য দুটি সূত্র আছে, যেটি **নিউমেরিকাল অ্যানালাইসিস** পদ্ধতি প্রয়োগ করে বের করা। সূত্র দুটি নিম্নরূপ –

$$m = \frac{\bar{x} \cdot \bar{y} - \bar{xy}}{(\bar{x})^2 - \bar{x}^2}$$

$$c = \bar{y} - m\bar{x}$$

এখন এই সূত্র দুটি টেবিল 3.2.2-এর ডেটার ওপরে ব্যবহার করে আমরা **m** ও **c -এর মান** নির্ণয় করে ফেলি চলুন :

X	Y	XY	X^2
6	350	2100	36
8	775	6200	64
12	1150	13800	144
14	1395	19530	196
18	1675	30150	324

টেবিল 3.2.2

এখন **m -এর মান** বের করার জন্য যা যা লাগবে, চলুন সেগুলো বের করে ফেলি –

$$\bar{X} = \frac{6 + 8 + 12 + 14 + 18}{5} = 11.6$$

$$\bar{Y} = \frac{350 + 775 + 1150 + 1395 + 1675}{5} = 1069$$

$$\bar{XY} = \frac{2100 + 6200 + 13800 + 19530 + 30150}{5} = 14356$$

অধ্যায় ৩ : লিনিয়ার রিগ্রেশন (Linear Regression)

$$(\bar{X})^2 = (11.6)^2 = 134.56$$

$$\bar{x}^2 = \frac{36 + 64 + 144 + 196 + 324}{5} = 152.8$$

সুতরাং

$$m = \frac{(11.6 \times 1069) - 14356}{134.56 - 152.8} = \frac{-1955.6}{-18.24} = 107.21$$

এখন m , \bar{y} ও \bar{x} মান (3) নম্বরে বসিয়ে পাই,

$$C = 1069 - (107.21 \times 11.6) = -174.6$$

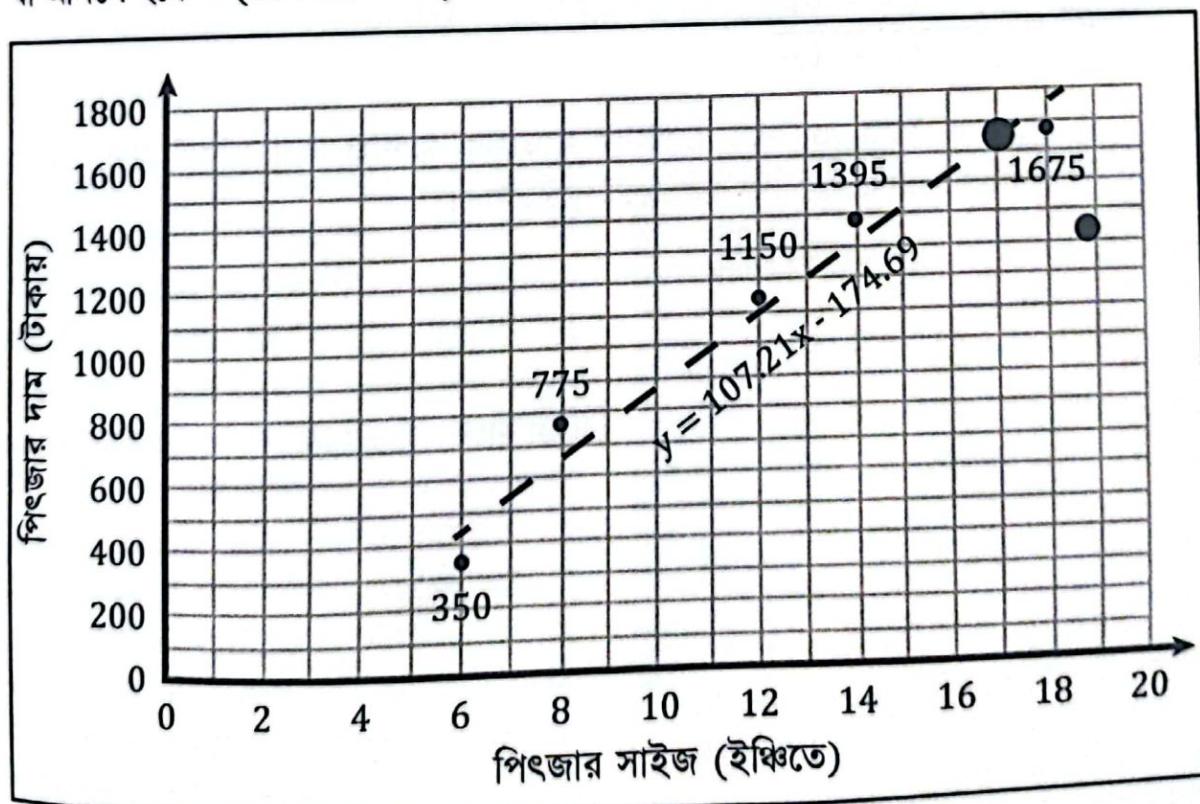
সুতরাং আমাদের কাঞ্চিত সরলরেখার সমীকরণ দাঁড়াল -

$$y = 107.21x - 174.6$$

এটি আমাদের রিগ্রেশন লাইন বা বেস্ট ফিট লাইন।

এখন এই রেখার সমীকরণে যদি আমরা $x = 17$ বসাই, তাহলেই আমরা পেয়ে যাব আমাদের 17 ইঞ্জিনিয়ার দাম,

যা আসলে হবে $= (107.21 \times 17) - 174.6 = 1647.96$ টাকা।



গ্রাফ 3.2.2 : পিংজার সাইজ ও দামের মধ্যে সম্পর্ক

এবং আমরা যদি এখন আমাদের গ্রাফ 3.2.1-এ আমাদের রিপ্রেশন লাইন এবং নতুন পাওয়া 17 ইঞ্জিনিয়ার দাম যোগ করি তাহলে আমরা গ্রাফ 3.2.2-এর মতো একটি গ্রাফ পাব।

গ্রাফটিতে আমরা ধূসর রঙের ডটেড যেই লাইনটি দেখতে পাচ্ছি, সেটিই আমাদের রিপ্রেশন লাইন। আর এই লাইনের ওপরের কালো রঙের একটি বড়ো বিন্দু দেখতে পাচ্ছি আমরা, সেটিই হলো আমাদের নতুন পাওয়া ডেটা পয়েন্ট - (17, 1647.96)।

এখানে আরেকটি বিষয় আপনাদের জানিয়ে রাখি। আপনার লিনিয়ার রিপ্রেশনের হাইপোথিসিস আসলে কতটুকু ভালো পারফরম করছে, অর্থাৎ আপনার হাইপোথিসিস কত ভালো ফিট করছে আপনার ডেটার ভেতরে, সেটি পরীক্ষা করার জন্য লিনিয়ার রিপ্রেশনের ক্ষেত্রে একটি মান বের করা হয়, যেটাকে বলা হয় আর-স্ক্যারড মান (R-Squared Value)। এর মান 0% থেকে 100%-এর মধ্যে যে-কোনো কিছু হতে পারে। এই R-Squared মান হচ্ছে একটি ভালো-মন্দের সূচকের মতো। কোনো হাইপোথিসিসের জন্য মান যত বেশি হবে, আমরা বুঝে নেব সেই হাইপোথিসিস মডেল তত ভালোভাবে ফিট করবে ডেটার ভেতর। এটি নিয়ে বিস্তারিত আলোচনা করা হয়েছে পরিচ্ছেদ ১৩.১-এ।

আমাদের এই সরলরেখার ক্ষেত্রে R-Squared Value-এর মান বের করার জন্য আমরা নিচের সূত্রটি ব্যবহার করব –

$$R^2 = \frac{\sum_{i=1}^n (\hat{y}_i - \bar{y})^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

এখানে \hat{y}_i , \bar{y} ও y_i হচ্ছে যথাক্রমে আমাদের মডেলের আন্দাজ করা প্রতিটি মান, সমস্ত আউটপুটের প্রকৃত মান এবং প্রতিটি প্রকৃত আউটপুটের মান।

আমাদের কাঙ্ক্ষিত সরলরেখার সমীকরণ ছিল –

$$y = 107.21x - 174.6$$

এখন আমরা এই সমীকরণ ব্যবহার করে x -এর বিভিন্ন মানের জন্য আমাদের লিনিয়ার মডেলের প্রেডিকশন বের করে ফেলতে পারি।

$$\begin{aligned} X = 6, & \quad y = 468.66 \\ X = 8, & \quad y = 683.08 \\ X = 12, & \quad y = 1111.92 \\ X = 14, & \quad y = 1326.34 \\ X = 18, & \quad y = 1755.18 \end{aligned}$$

মানগুলোকে R-Squared Value-এর সূত্রে বসিয়ে পাই,

অধ্যায় ৩ : লিনিয়ার রিগ্রেশন (Linear Regression)

Page 44

$$\checkmark R^2 = \frac{(468.88-1069)^2 + (683.08-1069)^2 + (1111.92-1069)^2 + (1326.34-1069)^2 + (1755.18-1069)^2}{(350-1069)^2 + (775-1069)^2 + (1150-1069)^2 + (1395-1069)^2 + (1675-1069)^2}$$
$$= \frac{360144.01 + 148934.26 + 1842.13 + 66223.88 + 470842.99}{516961 + 86436 + 6561 + 106276 + 367236}$$
$$= 0.96$$

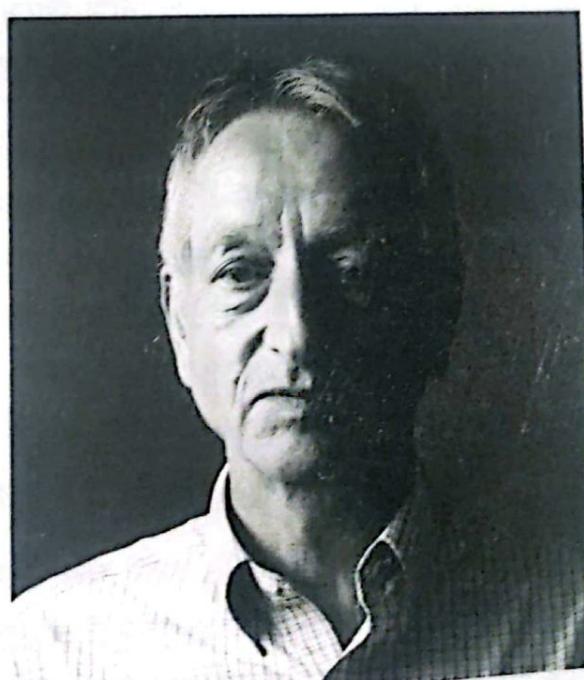
সব সময় আমাদের লক্ষ্য থাকবে, R-Squared মান ম্যাক্সিমাইজ (Maximize) করে এমন হাইপোথিসিস খুঁজে বের করা। অর্থাৎ m ও c -এর এমন মান বের করা যাব জন্য R-Squared মান 100% বা এর সর্বোচ্চ যতটুকু কাছাকাছি যাওয়া সম্ভব, ততটুকু যায় এবং আমরা নিউমেরিকাল অ্যানালাইসিস মেথড প্রয়োগ করে সরাসরি কীভাবে সেই m ও c -এর মান বের করা যায়, সেটি একটু আগেই দেখেছি।

আপনারা যদি এ পর্যন্ত মোটামুটি বুঝে থাকেন, তাহলে আপনারা এখন নিউমেরিকাল অ্যানালাইসিস মেথড প্রয়োগ করে লিনিয়ার রিগ্রেশন অনেকখানিই বুঝে গেছেন এবং এখন আপনারা এটি নিজেরাই হাতে কলমে কোড করতে পারবেন বলে আশা করছি।

তবে আর দেরি কেন, এখনই বসে পড়ুন কোড করতে।

পরিচ্ছন্দ ৩.৩ : গ্রেডিয়েন্ট ডিসেন্ট অ্যালগরিদম (Gradient Descent Algorithm)

গ্রেডিয়েন্ট ডিসেন্ট অ্যালগরিদমটি অনেক পুরোনো একটি অ্যালগরিদম। এর সূচনা হয় অগাস্টিন লুইস কসির (Augustin-Louis Cauchy; 1789-1857) হাত ধরে। তিনিই প্রথম *Comptes rendus de l'Académie des Sciences* জার্নালে এই গ্রেডিয়েন্ট ডিসেন্ট নামে নন লিনিয়ার অপটিমাইজেশন মেথডটি উল্লেখ করেন। কিন্তু পরবর্তীকালে জেফরি হিন্টন (Geoffrey Hinton, 1947) প্রথম মেশিন লার্নিংয়ে গ্রেডিয়েন্ট ডিসেন্ট অ্যালগরিদম ব্যবহার করেন। সুতরাং, জেফরি হিন্টনকেই মেশিন লার্নিংয়ে গ্রেডিয়েন্ট ডিসেন্ট-এর জনক বলা চলে।



ছবি 3.3.1 : জেফরি হিন্টন (Geoffrey Hinton, 1947)

গ্রেডিয়েন্ট ডিসেন্ট অ্যালগরিদম ব্যবহার করে লিনিয়ার রিগ্রেশন করার আগে আমাদের একটু বুঝে নিতে হবে যে গ্রেডিয়েন্ট ডিসেন্ট অ্যালগরিদম আসলে কী এবং কীভাবে কাজ করে।

একটু আগে আমরা কতগুলো সূত্রে মান বসিয়ে সরাসরি হিসাব কয়ে m ও c -এর মান বের করেছি, যা মোটামুটি সঠিক। এখন চলুন, আমরা এই সূত্র দিয়ে মান বের না করে অন্য একটি পদ্ধতিতে চিন্তা করি। ধরা যাক, আমাদের যেহেতু m ও c -এর মান জানা নেই, তাই আমরা **প্রথমেই আন্দজে** (randomly) আমাদের m ও c -এর দুটো মান ধরে নিই। এটি আপনারা যে-কোনো কিছু ধরতে পারেন। আমি ধরে নিলাম $m = 50, c = 100$ । আপনারা অন্য কোনো মান ধরতে পারেন, তাতে কোনো অসুবিধা নেই।

তাহলে, এখন আমাদের সরলরেখার সমীকরণ দাঁড়াল $y = 50x + 100$ এবং এটিতে আমাদের জানা x -এর মানগুলো থেকে যে-কোনো একটি যদি বসাই (যেমন, $x = 6$), তাহলে আমরা পাই $y = 400$ । এটি কিন্তু আমাদের আসল দামের সঙ্গে মিলল না, কেননা আসল দাম দেওয়া আছে 350 টাকা। এখানে যেটি হলো, আমরা আসলে শুরুতে ভুল আন্দজ করেছিলাম m ও c -এর মান, যার কারণে আমরা $400 - 350 = 50$ টাকার মতো গরমিল পাচ্ছি।

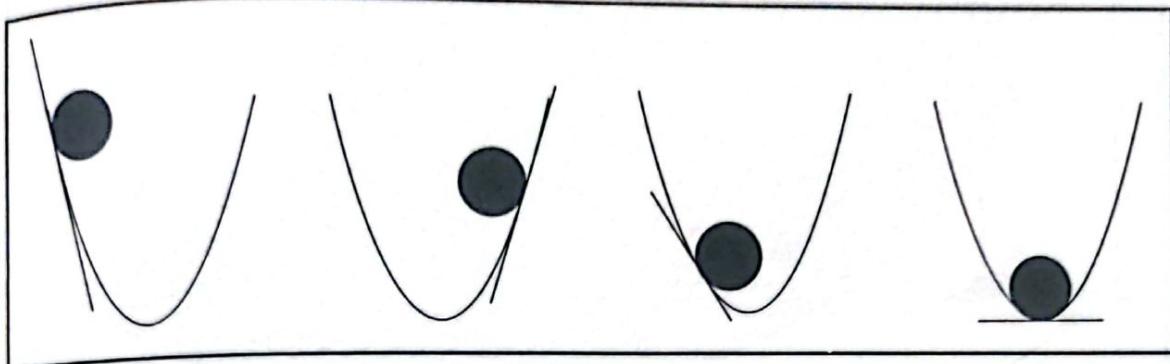
এজন্য আমাদের এখন যে কাজটি করতে হবে তা হলো, m ও c -এর মান **পরিবর্তন বা আপডেট** করতে হবে যাতে আমাদের **এই ভুল** ($\text{error} = 50$ টাকা গরমিল)-এর পরিমাণ কমে আসে। আমাদের লক্ষ্য হলো ভুলের পরিমাণ শূন্য বা এর কাছাকাছি নামিয়ে নিয়ে আসা। অর্থাৎ m ও c -এর মান আপডেট করতে করতে আমাদের ততক্ষণ পর্যন্ত আপডেট করতে হবে যতক্ষণ পর্যন্ত না আমাদের জানা প্রতিটি **ট্রেনিং ডেটার জন্য** এই ভুলের পরিমাণ শূন্য বা শূন্যের যত কাছাকাছি নামিয়ে আনা যায়, তত কাছাকাছি নেমে আসে। এই প্রতিবার মান আপডেট করে করে ভুল (বা Cost বা Error)-এর মান নামিয়ে নিয়ে আসার পদ্ধতিই হচ্ছে **গ্রেডিয়েন্ট ডিসেন্ট অ্যালগরিদম**।

একে গ্রেডিয়েন্ট ডিসেন্ট (Gradient Descent) বলার একটি কারণ আছে। গ্রেডিয়েন্ট হলো সোজা করে বলতে গেলে ঢাল। এখন প্রশ্ন হচ্ছে কীসের ঢাল? কোনো একটি গ্রাফে যদি কোনো একটি ফাংশন প্লট করা হয়, তাহলে সেই গ্রাফের যে-কোনো একটি বিন্দুতে একটি স্পর্শক আঁকলে, সেটি X অক্ষের সঙ্গে যে কোণ উৎপন্ন করে থাকে তাঁর ট্যানজেন্ট (Tangent)-এর মানকে বলে গ্রেডিয়েন্ট। আর ডিসেন্ট শব্দের মানে হলো কমানো বা কমে যাওয়া। তাঁর মানে গ্রেডিয়েন্ট ডিসেন্ট-এর মানে অনেকটা এরকম দাঁড়ায় যে ঢালের মান কমানো এবং কমাতে কমাতে শূন্যের কাছাকাছি নিয়ে আসা। শূন্যতে নিয়ে যেতে পারলে আরো ভালো।

গ্রেডিয়েন্ট ডিসেন্ট আসলে কী, সেটা বুঝতে ছবি 3.3.2 লক্ষ্য করুন। ছবিটি একটি 2D গ্রাফ। আমরা আমাদের পিংজার উদাহরণে দুটি প্যারামিটার বা ফিচার ভ্যালু নিয়ে কাজ করেছিলাম, m ও c । এই দুটির মান পরিবর্তন করে করে আমরা অবশ্যে একটি বেস্ট ফিট লাইন পেয়েছিলাম, যেটি আমাদেরকে ভুলের মান সর্বনিম্ন দিত।

অধ্যায় ৩ : লিনিয়ার রিফ্রেশন (Linear Regression)

এখন প্রেডিয়েন্ট ডিসেন্ট বোৰাৰ সুবিধার্থে, আপাতত আমৱা ধৰে নিই আমাদেৱ $c = 1$ । এটি পৱিবৰ্তিত হবে না, একই থাকবে। পৱিবৰ্তন হবে শুধু m -এৰ মান। এখন আমৱা যদি এৱকম একটি ফাংশন চিত্তা কৱি, যা ইনপুট হিসেবে নেবে m -এৰ মান এবং আউটপুট হিসেবে দেবে এৱে কিংবা কস্ট-এৰ মান; এবং আমৱা এই মানগুলো দিয়ে যদি একটি 2D গ্ৰাফ আঁকি, যেখানে X অক্ষ বৱাৰ নেব m এবং Y অক্ষ বৱাৰ নেব Cost, তাহলে আমৱা নিচেৱ 2D গ্ৰাফগুলোৱ মতো একটি গ্ৰাফ পাব (গ্ৰাফ 3.3.2)। আমৱা এখানে ৪টি সেৱকম 2D গ্ৰাফ নিয়েছি।

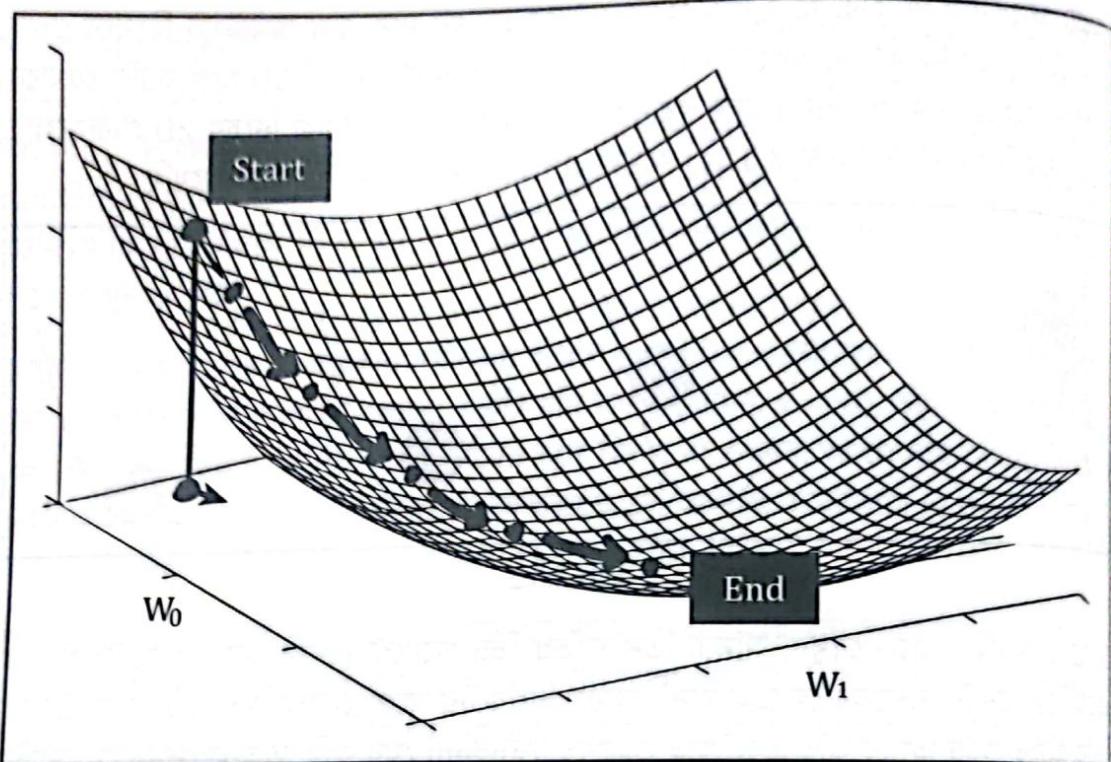


গ্ৰাফ 3.3.2

1, 2 ও 3 নম্বৰ গ্ৰাফে দেখুন, আমৱা তিনটি ভিন্ন ভিন্ন পয়েন্টে (গোল কালো পয়েন্ট) স্পৰ্শক এংকেছি। তিনটি স্পৰ্শকেৰ ক্ষেত্ৰেই যদি আমৱা স্পৰ্শকটিৱ সঙ্গে X -অক্ষ কত ডিগ্ৰি কোণ উৎপন্ন কৱে আছে সেটি হিসাব কৱি এবং সেই কোণেৰ Tangent বেৰ কৱি, তাহলে দেখব কোনোটিৰ মানই শূন্য হয় না। তাৰ মানে m -এৰ এই মানগুলোৱ জন্য আসলে ভুলেৰ মান সৰ্বনিম্ন আসবে না। কিন্তু 4 নম্বৰ গ্ৰাফে দেখুন, আমৱা m -এৰ যে মান নিয়েছি, সেই ডেটাপয়েন্টে যদি আমৱা স্পৰ্শক আঁকি, তাহলে সেটি X -অক্ষেৰ সমান্তৰাল হয়। তাৰ মানে X -অক্ষেৰ সঙ্গে স্পৰ্শকেৰ কোণেৰ মান 0 ডিগ্ৰি। অৰ্থাৎ, ঢাল হবে $\tan(0) = 0$ । অৰ্থাৎ, আমৱা ঢালেৰ মান সৰ্বনিম্নে নামিয়ে আনতে পৱেছি, তাৰ মানে এই পয়েন্টে কস্ট-এৰ মানও সৰ্বনিম্ন। কস্ট-এৰ মান 0 হবে – এৱকম কোনো কথা নেই। কস্ট-এৰ মান সৰ্বনিম্ন হলেই হলো। আমৱা তাই এভাৱে, শুৱতে আন্দাজে (Randomly) m -এৰ যে-কোনো একটি মান ধৰে নেব, এৱে পৱে ধাপে m -এৰ মান আপডেট কৱতে কৱতে 4 নম্বৰ গ্ৰাফেৰ মতো অবস্থায় নিয়ে আসব, যাতে আমৱা সৰ্বনিম্ন কস্ট পাই।

আশা কৱি সবাই কিছুটা হলেও প্ৰেডিয়েন্ট ডিসেন্ট অ্যালগৱিদমেৰ ধাৰণা বুৰাতে পৱেছেন। এখানে আৱেকটি গ্ৰাফ দেওয়া হলো (গ্ৰাফ 3.3.3)। এটি একটি 3D গ্ৰাফ। আগেৰ বাব আমৱা c -এৰ মান ঠিক রেখে শুধু m -এৰ মান পৱিবৰ্তন কৱেছিলাম। এবাৱ আৱ তা নয়। m ও c দুটিই পৱিবৰ্তিত হবে। তাৰ মানে আমাদেৱ ফিচাৰ দাঁড়াল দুটি। আৱ সেই সঙ্গে আছে কস্ট। তাই আমাদেৱ গ্ৰাফটি হচ্ছে 3D গ্ৰাফ। এই গ্ৰাফেৰ X -অক্ষ ও Y -অক্ষ বৱাৰ আমৱা নিয়েছি m ও c যাদেৱকে এখানে W_0 ও W_1 হিসেবে চিহ্নিত কৱা হয়েছে। এৱে বিপৰীতে, কস্ট বা এৱে (error) প্ৰট কৱা হয়েছে Z -অক্ষ বৱাৰ, যাকে $E[W]$ দ্বাৱা চিহ্নিত কৱা হয়েছে। আমাদেৱ লক্ষ্য হচ্ছে

আবারও আগের মতোই গ্রাফের সর্বনিম্ন বিন্দুতে পৌছানো, যেইখানে স্পর্শক অংকলে তাঁর চাঙ্গ এর মান শূন্য হবে।



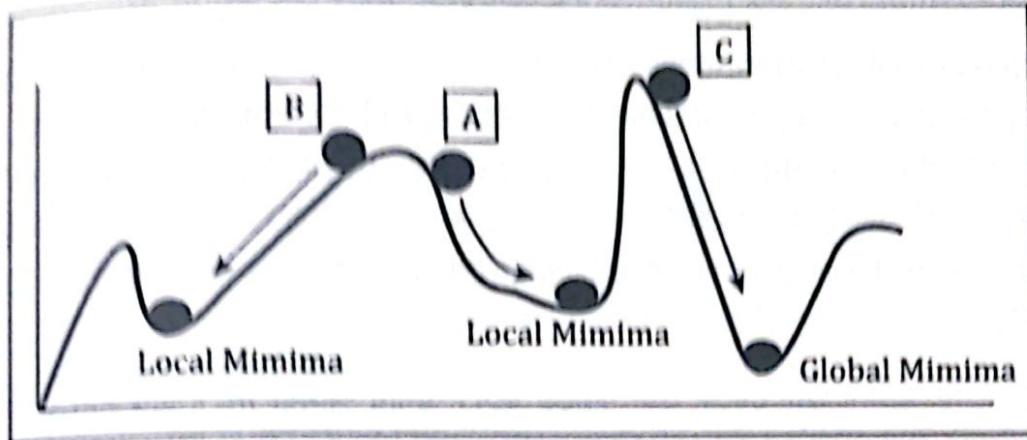
গ্রাফ 3.3.3

আমরা আবারও m ও c -এর ভ্যালু আগের মতোই আন্দাজে কোনো একটি মান দিয়ে শুরু করব, তারপর দুটির মানই আপডেট করতে থাকব, যতক্ষণ পর্যন্ত না গ্রাফের সর্বনিম্ন অবস্থানে পৌছাই। এটিই প্রেডিয়েন্ট ডিসেন্ট। গ্রাফ 3.3.3-তে দেখুন, আমরা Start লেখা পয়েন্ট থেকে শুরু করেছিলাম এবং দেখুন পয়েন্টটি কিন্তু আমাদের গ্রাউন্ড (এখানে গ্রাউন্ড বলতে আসলে গ্রাফ 3.3.3-এ দেখানো $W_0 - W_1$ তল বোঝানো হয়েছে) থেকে অনেক উঁচুতে, অর্থাৎ error অনেক বেশি। এরপর আমরা আস্তে আস্তে মান আপডেট করতে করতে End পয়েন্টে এসে পৌছেছি, যেটি একেবারে গ্রাউন্ডে, অর্থাৎ error-এর মান শূন্য। একটু ভালো করে লক্ষ করবেন, Start থেকে End-এ যাওয়ার ব্যাপারটি এখানে ছবি দেখে অনেকটা চাদরের গা বেয়ে স্লাইড করে নামার মতো মনে হচ্ছে এবং ব্যাপারটি খুবই মজার। ব্যাপারটি ঘটেও অনেকটা এরকমভাবেই। এই চাদরের মতো তলটিকে বলা হয় এরর প্লেন (Error Plane)।

এখানে আরেকটি বিষয় উল্লেখ্য। একটি গ্রাফে অনেকগুলো সর্বনিম্ন বিন্দু (Minimum Point) থাকতে পারে এবং সবগুলো বিন্দুতেই স্পর্শকের ঢালের মান শূন্য আসতে পারে। সেসব বিন্দুকে আমরা বলব লোকাল মিনিমা (Local Minima)। আর লোকাল মিনিমার মধ্যে যে লোকাল

অধ্যায় ৩ : লিনিয়ার রিজেসন (Linear Regression)

ত্বরিত হিস্ত পর্যবেক্ষণ, তাকে বলল গ্লোবাল মিনিমাম (Global Minimum), যদি একাধিক গ্লোবাল মিনিমাম থাকে, তাহলে তাদের মধ্যে গ্লোবাল মিনিমা (Global Minima)। নিচের ছবিটি দেখলে ধারণাটি আরেকটি পরিষ্কার হবে।



গ্রাফ 3.3.4

গ্রাফ 3.3.4-এ দেখুন, মোট তিনটি মিনিমাম পয়েন্ট আছে, এর মধ্যে একেবারে ডানে যেটি, সেটিই হচ্ছে আমাদের গ্লোবাল মিনিমাম, বাকি দুটি লোকাল মিনিমা। আমাদের সব সময় লক্ষ্য থাকবে যে, আমরা গ্লোবাল মিনিমাম পয়েন্টে পৌছাব।

একটি জিনিস লক্ষ্য করবেন, আমি যখন গ্রেডিয়েন্ট ডিসেন্ট অ্যালগরিদম ব্যবহার করছি, তখন আমি শুধু এর প্লেন বেয়ে শুধু নিচের দিকে যেতে পারব, ওপরের দিকে কখনোই উঠতে পারব না। এর কারণ হচ্ছে, আমরা চাইছি কস্ট/এর মিনিমাইজ করতে, তাই না? আমরা যদি এর প্লেন বেয়ে ওপরের দিকে উঠি, তাহলে কিন্তু আমাদের এর বাড়তে থাকবে, যা আমাদের জন্য Allowed নয়। তাই, আমরা শুধু এর প্লেন বেয়ে নিচের দিকেই যেতে পারব, ওপরের দিকে উঠতে পারব না।

এখন ধরুন, আমরা এমনভাবে আমাদের m, c -এর মান ধরেছি, যে আমরা এর প্লেনে A অবস্থানে আছি। যদিও আমাদের এর প্লেনের গ্রাফটি 3D গ্রাফ ছিল, এখানে বোঝানোর সুবিধার জন্য 2D আকারে দেখাচ্ছি। আমাদের লক্ষ্য হচ্ছে A পয়েন্ট থেকে একেবারে ডান দিকে থাকা গ্লোবাল মিনিমাম পয়েন্টে আমাদের পৌছাতে হবে। এজন্য আমরা যেহেতু শুধু নিচের দিকে যেতে পারব, তাই ডান দিকের ঢাল বেয়ে নিচের দিকে নামতে থাকি (অনেকটা পাহাড় থেকে নামার মতো)। নামতে নামতে একসময় দেখুন আমরা একটি লোকাল মিনিমাম পয়েন্টে এসে আটকা পড়ে যাব।

কেবলা, আমাদের যদি গ্লোবাল মিনিমামে পৌছাতে হয়, তাহলে আমাদের লোকাল মিনিমান পার হয়ে পাহাড় বেয়ে ওপরে উঠে আবার নিচে নামতে হবে, তবেই আমরা সেখানে পৌছাতে পারব। কিন্তু, পাহাড় বেয়ে ওপরে উঠা আমাদের জন্য Allowed নয়, সেটি আগেই বলা হয়েছে। তাহলে

ব্যাপারটি কী দাঁড়াল? আমরা লোকাল মিনিমান পয়েন্টে আটকা পড়ে গেলাম এবং এখান থেকে আমরা আর বের হতে পারব না, আমাদের অ্যালগরিদম এটিকেই আমাদের সমাধান হিসেবে রিটার্ন করবে যদিও এটি মোটেই গ্লোবাল মিনিমাম নয়। এই ধরনের পরিস্থিতিকে বলা হয় 'Stuck in Local Minimum', যেটি মেশিন লার্নিংয়ের ক্ষেত্রে খুবই সাধারণ একটি সমস্যা।

এর সমাধানটি খুবই সহজ। আমাদের আবার গোড়া থেকে পুরো অ্যালগরিদম শুরু করতে হবে এবং এবারও র্যান্ডমলি m, c -এর অন্য আরেকটি মান ধরে নিতে হবে। এবার ধরা যাক, আমরা র্যান্ডমলি আবার B অবস্থান থেকে শুরু করলাম। এবারও দেখুন, আমরা যেহেতু কেবল নিচের দিকে যেতে পারি, তাই আমরা বাঁ দিকে যেতে থাকব, যার ফলে আমরা ধাপে ধাপে গ্লোবাল মিনিমাম থেকে দূরে সরে যেতে থাকব এবং একসময় একটি লোকাল মিনিমান পয়েন্টে আটকা পড়ে যাব।

তখন আমাদের আবার শুরু থেকে শুরু করতে হবে। এবার আমরা m, c -এর মান র্যান্ডমলি নিয়ে, ধরা যাক, C অবস্থান থেকে শুরু করি, তাহলে দেখুন এবার কিন্তু আমরা ডান দিক দিয়ে নিচে নামতে নামতে গ্লোবাল মিনিমাম পয়েন্টে পৌঁছে যাব। এই পদ্ধতিকে বলা হয় Random Restart এবং এটি ততক্ষণ পর্যন্ত আমরা করতে থাকব যতক্ষণ পর্যন্ত না আমরা গ্লোবাল মিনিমান পয়েন্টে পৌঁছাচ্ছি কিংবা আমাদের ইটারেশন থ্রেশহোল্ড (Iteration Threshold – সর্বোচ্চ কর্তব্য Global Minimum খুঁজে না পাওয়া পর্যন্ত Random Restart করতে পারব তার একটি মান নির্ধারিত করে দেওয়া থাকে) অতিক্রম করে যাচ্ছি।

এখন কথা হচ্ছে, অ্যালগরিদমের তো জানার কথা নয় যে Global বা Local Minimum-এর মান কত, তাহলে সে কেমন করে বুঝবে যে সে মিনিমামে পৌঁছেছে কি না। এর উত্তর হচ্ছে, আমরা প্রতিবার লুপ ঘুরে আসার পরে যাচাই করে দেখব যে, আমরা বর্তমান m, c -এর জন্য Error-এর যে মান পেলাম, সেটি আগে যতবার লুপ ঘুরেছে প্রতিবারের পাওয়া সব Error-এর মানের মধ্যে সর্বনিম্ন কি না, যদি হয় তাহলে এটি মিনিমাম মান হিসেবে সংরক্ষণ করব, আর যদি না হয়, তাহলে এবারের আগ পর্যন্ত মিনিমাম মান যা ছিল তা-ই থাকবে এবং পরের বার লুপ চালাব। এভাবে একসময় লুপ চলতে চলতে যদি আমরা দেখি যে বেশ অনেকবার লুপ চলেছে আর আমাদের থ্রেশহোল্ডও পূর্ণ হয়ে গেছে কিন্তু Error-এর মান আর কমছে না, তখন ধরে নেব Error-এর মান এর চেয়ে আর কম সম্ভব নয়। তখন তাকেই আমরা সর্বনিম্ন মান হিসেবে ধরে নেব।

এখন, এই সর্বনিম্ন মান Global/Local যে-কোনোটিই হতে পারে। তাই, তাকে আবার যাচাই করে নেওয়ার জন্য Randomly আরো বেশ কয়েকবার Restart করে পুরো লুপ চালাব। ধরুন আমি 3 বার Random Restart করলাম, প্রতিবার আমি 10 বার করে লুপ চালালাম। প্রথম Random Restart-এ আমি এরবের মান পেলাম ধরা যাক 10, দ্বিতীয়বার পেলাম 8 এবং তৃতীয়বার পেলাম 4। সুতরাং, এই তিনটির মধ্যে সর্বনিম্ন হচ্ছে 4, আর তাই তৃতীয় Random

Restart-এ, যে m , c -এর জন্য আমরা error হিসেবে ৪ পেয়েছি, সেই m , c -এর মানই হবে আমার বেস্ট ফিট লাইনের m , c -এর মান।

এই হচ্ছে গ্রেডিয়েন্ট ডিসেন্ট অ্যালগরিদম সম্পর্কে যতটুকু জানার দরকার তার বিস্তারিত। পরবর্তী সময়ে আমরা দেখব কীভাবে গ্রেডিয়েন্ট ডিসেন্ট অ্যালগরিদম বিভিন্ন মেশিন লার্নিং পদ্ধতিতে ব্যবহার করতে হয়।

পরিচ্ছেদ ৩.৪ : গ্রেডিয়েন্ট ডিসেন্ট (Gradient Descent) অ্যালগরিদম ব্যবহার করে লিনিয়ার রিপ্রেশন

আমরা প্রথমে দুটি ফাংশন নির্ধারণ করব, একটি হচ্ছে কস্ট ফাংশন (Cost Function), আরেকটি অবজেক্টিভ ফাংশন (Objective Function)। আমাদের প্রথমে বুঝতে হবে এই দুটি ফাংশন আসলে কী।

ধরা যাক, আমার বয়স 20 বছর। আমি একটি প্রোগ্রাম লিখলাম, যার কাজ হচ্ছে আমার বয়স আন্দাজ করা এবং যদি সে আমার ঠিক বয়স বলতে পারে, তাহলে প্রোগ্রাম বন্ধ হয়ে যাবে। আমার আসল বয়সকে যদি আমি A দিয়ে লিখি এবং আমার আন্দাজ করা বয়সকে যদি আমি X দিয়ে লিখি, তাহলে একটি কস্ট ফাংশন এরকম হতে পারে $F(X) = X - A$ । এটি সরাসরি আমার আসল বয়স এবং প্রোগ্রামের দ্বারা আন্দাজ করা বয়সের পার্থক্য রিটোর্ন করবে।

আর অবজেক্টিভ ফাংশন হচ্ছে, যে ফাংশনটিকে আমরা অপটিমাইজ করতে চাইছি, সেটিই। অর্থাৎ, আমাদের এই ক্ষেত্রে, আমরা চাইছি যে আমাদের কস্ট কমিয়ে আনতে, অর্থাৎ X -এর এমন মান আন্দাজ করতে চাইছি, যাতে সেটি আমার আসল বয়সের সমান হয়, ফলে কস্ট-এর মান শূন্য হয়। সাধারণত মেশিন লার্নিংয়ে এই কস্ট ও অবজেক্টিভ ফাংশন একই অর্থে ঘূরিয়ে-ফিরিয়ে ব্যবহার করা হয়।

কস্ট ফাংশন অনেক ধরনের হতে পারে। একেকটি ব্যবহারের একেক রকম সুবিধা। আমরা লিনিয়ার রিপ্রেশনের ক্ষেত্রে সাধারণত যে ফাংশনটি কস্ট ফাংশন হিসেবে ব্যবহার করব, তার নাম Squared Error Function। এটি খুবই জনপ্রিয় এবং খুব সন্তুষ্ট সবচেয়ে বেশি ব্যবহৃত একটি কস্ট ফাংশন।

কস্ট ফাংশনটি লেখার আগে আমরা একটি কাজ করব, তা হলো, c ও m -কে এখন থেকে আমরা w_0 ও w_1 দিয়ে প্রকাশ করব। এই w_0 , w_1 ইত্যাদিকে ওয়েইট (Weight) বলা হয়। এর কারণ হচ্ছে, এদেরকে যে রাশির সঙ্গে সহগ বা কো-এফিসিয়েন্ট (Co-efficient) হিসেবে ব্যবহার করা হয়, এরা সেই সংখ্যার গুরুত্ব প্রকাশ করে।

আর আমাদের পরিচ্ছেদ ৩.২-এ ব্যবহৃত হাইপোথিসিসের সমীকরণে আমরা ফাইনাল আউটপুট বোঝানোর জন্য Y ব্যবহার করেছিলাম, এখন Y -এর পরিবর্তে $h_w(x)$ ব্যবহার করব। $h_w(x)$ দিয়ে বোঝাবে আমরা x ডেটা পয়েন্টের জন্য কাজ করছি এবং আমাদের প্যারামিটার (চাল ও y -intercept) হচ্ছে w সেটের অন্তর্ভুক্ত (w_0 ও w_1)। আমরা এই কাজটি করব আমাদের কিছু গাণিতিক সমীকরণ লেখার সুবিধার্থে। অর্থ একই থাকবে, সমীকরণ একই থাকবে, শুধু প্রকাশকারী চিহ্নগুলো পরিবর্তন করলাম মাত্র।

তাহলে আমাদের হাইপোথিসিসের সমীকরণের চেহারা দাঁড়ায়,

$$h_w(x) = w_0 + w_1 x$$

তার মানে $h_w(x)$ -এর মান হচ্ছে আমাদের প্রেডিকটেড (Predicted) বা অনুমিত মান, যা কিনা আমরা w_0 ও w_1 -এর মান বের করার মাধ্যমে হিসাব করে বের করছি। আর Y -কে যদি আমরা আমাদের আসল আউটপুটের মান ধরে নিই, যেটি আমাদের লেবেলড ট্রেনিং ডেটা হিসেবে দেওয়া আছে, তাহলে আমাদের এর কিংবা কস্ট ফাংশন-এর মান প্রাথমিক রূপ দাঁড়ায়,

$$h_w(x) - Y$$

এখন দেখুন, আমরা যদি আমাদের কস্ট ফাংশনকে এমন একটি অবজেকটিভ ফাংশন হিসেবে লেখি, যাতে আমাদের কস্ট ফাংশনকে মিনিমাইন করতে হয়, তাহলে,

$$\text{Cost} = h_w(x) - Y$$

এ ক্ষেত্রে, $h_w(x)$ -এর মান যত বেশি $-\infty$ (negative infinity)-এর দিকে যাবে, অর্থাৎ যত বেশি ঋণাত্মক হবে, তত কস্ট ফাংশনের মান মিনিমাইজড হবে। কিন্তু দেখুন, শুধু কস্ট ফাংশন মিনিমাইজ করলেই কি আমাদের কাজ হাসিল হচ্ছে? আমাদের লক্ষ্য হচ্ছে আমাদের আন্দাজ করা মান $h_w(x)$ ও আসল মান Y -এর সমান করা, কিংবা যতটা পারা যায় কাছাকাছি নিয়ে আসা এবং তার মাধ্যমে কস্ট ফাংশনকে মিনিমাইজ করা। কিন্তু যদি, আমরা শুধু $h_w(x)$ -এর মান $-\infty$ -এর দিকে নিতে থাকি, তাহলে কস্ট ফাংশনের মান মিনিমাইজ হচ্ছে ঠিকই, কিন্তু দুটি মান কাছাকাছি আসবে বা সমান হবে, এরকম কোনো বাধ্যবাধকতা (Constraint) তৈরি হচ্ছে না।

তাই, আমরা কস্ট ফাংশন হিসেবে শুধু এর ফাংশন না নিয়ে স্কয়ারড এর ফাংশন (Squared Error Function) নেব,

more filter $\{h_w(x) - y\}^2$ cause squ ৬ error বাট্টে যদি $h(w)$ পার্শ্বে \leftarrow করা থাএ

এর ফলে একটি সুবিধা হবে যে, এই ফাংশনের মান তখনই কেবল মিনিমাইজ হবে যখন $h_w(x)$ ও Y সমান বা খুব কাছাকাছি হবে। যদি $h_w(x)$ ও Y -এর মান সমান হয়, তাহলে, কস্ট ফাংশনের মান হবে শূন্য, যেটি এর সর্বনিম্ন মান। $h_w(x)$ -এর মান আমরা এখন যতই ঋণাত্মক বানাই না

অধ্যায় ৩ : লিনিয়ার রিপ্রেশন (Linear Regression)

কেন, শেষমেশ সেটি বর্গ হয়ে একটি বড়ো ধনাত্মক মানে পরিণত হবে, যেটি আমাদের কস্ট ফাংশন মিনিমাইজ করার লক্ষ্য পরিপূর্ণ করবে না।

এভাবে Squared Error Function ব্যবহার করলে যেহেতু আমরা দুটো মান সমান হওয়ার বাধ্যবাধকতাও দিতে পারছি এবং সেই সঙ্গে কস্ট ফাংশনও মিনিমাইজ করতে পারছি, এ কারণে Squared Error ফাংশনকেই আমরা কস্ট ফাংশন হিসেবে ব্যবহার করব।

এখন, কস্ট ফাংশন হিসেবে যদি শুধু $\{h_w(x) - y\}^2$ রাখিটি নিই, তাহলে আমরা যে কস্ট পাব, সেটি হচ্ছে আমাদের একটিমাত্র সিংগেল ডেটা পয়েন্ট কিংবা ট্রেনিং ডেটা x (একটিমাত্র পিংজার সাইজ)-এর জন্য। কিন্তু আমাদের লক্ষ্য হচ্ছে সবগুলো ট্রেনিং ডেটার জন্য আমাদের একটি সামগ্রিক কস্ট হিসাব করা। সেটি যদি করতে হয়, তাহলে আমাদের সব ডেটা পয়েন্টের জন্য এরের হিসাব করে তাকে মোট ট্রেনিং ডেটার সংখ্যা দিয়ে ভাগ করে গড় মানটি নিতে হবে। যদি তা-ই করি, তাহলে আমাদের কস্ট ফাংশনের সর্বশেষ চেহারা দাঁড়াবে :

$$\checkmark = \frac{1}{2m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}^2$$

এখানে $x^{(i)}, y^{(i)}$ মানে হচ্ছে i -তম ট্রেনিং ডেটা এবং m হচ্ছে মোট ট্রেনিং ডেটার সংখ্যা। এ ছাড়া এখানে একটি অতিরিক্ত 2 দিয়ে ভাগ করা হয়েছে কিছু গাণিতিক হিসাবের সুবিধার্থে।

কস্ট ফাংশনটিকে আমরা যদি $J(w_0, w_1)$ দিয়ে প্রকাশ করি (শুধুই একটি গাণিতিক প্রকাশ), তাহলে আমাদের ফাংশনের সর্বশেষ রূপ দাঁড়ায় এরকম –

$$\checkmark J(w_0, w_1) = \frac{1}{2m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}^2$$

এটিই আমাদের কাঞ্জিক্ত কস্ট ফাংশন। আমাদের লক্ষ্য হচ্ছে এই ফাংশনটির প্যারামিটার w_0 ও w_1 -এর এমন কোনো একটি মান বের করা, যার জন্য $J(w_0, w_1)$ -এর মান সর্বনিম্ন নামিয়ে নিয়ে আসা (মিনিমাইজ করা) যায়। এটিই আমাদের অবজেকটিভ ফাংশন এবং এটিকে লেখা যায়

এভাবে :

$$\text{Objective Function} = \underset{w_0, w_1}{\text{minimize}} J(w_0, w_1)$$

$$= \underset{w_0, w_1}{\text{minimize}} \frac{1}{2m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}^2$$

এখন আমাদের এই লিনিয়ার রিপ্রেশন চালানোর জন্য গ্রেডিয়েন্ট ডিসেন্টের ধাপগুলো দাঁড়াছে :

এখন আমাদের এই লিনিয়ার রিপ্রেশন চালানোর জন্য গ্রেডিয়েন্ট ডিসেন্টের ধাপগুলো দাঁড়াছে :

- প্রথমে w_0 ও w_1 -কে দুটি র্যানডম মান অ্যাসাইন করি (আপনারা দুটির মানই 1 দিয়ে শুরু করতে পারেন)।

- এরপর w_0 ও w_1 -এর মান প্রতি ইটারেশনে একবার করে আপডেট করতে হবে এমনভাবে যাতে $J(w_0, w_1)$ -এর মান ক্রমশ কমে আসতে থাকে।
- একসময় দেখা যাবে 5-10 ইটারেশনের পরেও $J(w_0, w_1)$ -এর মান আর উল্লেখযোগ্য হারে কমছে না। অর্ধাং লুপের ইটারেশন সংখ্যার বিপরীতে $J(w_0, w_1)$ প্ট করলে, দেখা যাবে ইটারেশনের সংখ্যা একটি মান অতিক্রম করার পর গ্রাফ এক জায়গায় প্রায় সরলরেখিক হয়ে গেছে। হয়তো 4-5 বার লুপ ঘোরালে 0.001 কমছে কিংবা এরকম শুধু অল্প পরিমাণে কমছে। তখন আপনি আপনার লুপ চালানো শেষ করে দিতে পারেন।
- লুপ শেষ হয়ে গেলে আপনি যে w_0 ও w_1 -এর মানগুলো পেলেন, এটিই এদের অপটিমাল ভ্যালু, যার জন্য আপনি $J(w_0, w_1)$ -এর মান সর্বনিম্ন পাচ্ছেন।
- এখন w_0 ও w_1 মান আপনি আপনার হাইপোথিসিস $h_w(x) = w_0 + w_1x$, সমীকরণে বসালেই আপনার কান্ডিক্ষত হাইপোথিসিস পেয়ে যাবেন। এখন এতে কোনো অজানা x -এর মান বসালে আমরা তার জন্য কান্ডিক্ষত আউটপুট পেয়ে যাব।

সবশেষে যেটি বাকি থাকে, সেটি হচ্ছে আমরা w_0 ও w_1 -এই প্যারামিটার দুটির মান আপডেট কীভাবে করব? এই আপডেটের অংশটুকুই হচ্ছে আমাদের কাজের মূল অংশ।

এর জন্য সোজাসাপটা সূত্র হচ্ছে –

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(w_0, w_1)$$

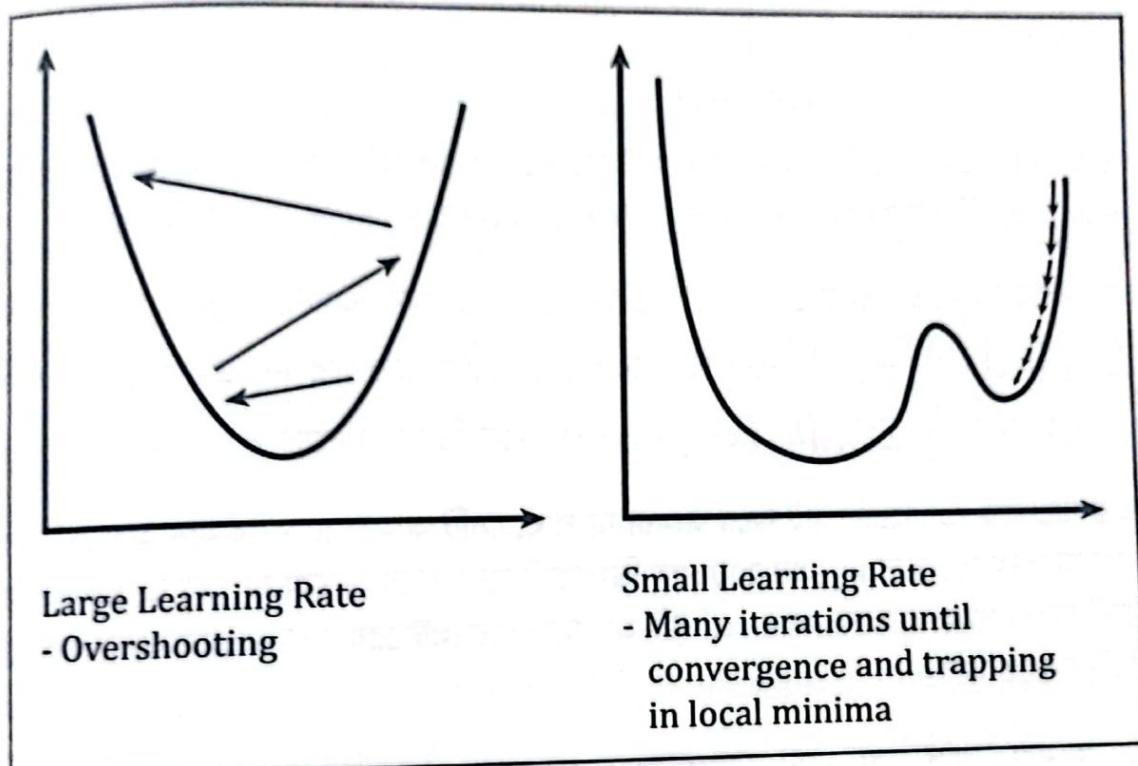
দেখতে অনেক নিরীহ মনে হলেও, আসলে অতটা সরল নয় এটি। কিছু জটিলতা আছে এর মধ্যে। সমীকরণটির প্রতিটি অংশের কোনটির কী অর্থ, তাহলে সেটি একটু জেনে নেওয়া যাক।

প্রথমেই জেনে নিই, সমীকরণটি ব্যবহৃত হচ্ছে j -এর সব মানের জন্য। এখানে j -এর মান হচ্ছে 0, 1 ইত্যাদি যা আমরা w_0, w_1 ইত্যাদি লেখার সময় সাবসক্রিপ্ট হিসেবে ব্যবহার করেছি। আমরা যদি আরো একটি ভ্যারিয়েবল নিতাম (হয়তো পিংজার সাইজের পাশাপাশি অন্য কোনো একটি ফিচার) এবং তাকে হয়তো x_2 দিয়ে প্রকাশ করে তাঁর জন্য ওয়েইট w_2 ব্যবহার করতাম, তাহলে j -এর মান হতো 0, 1, 2 ইত্যাদি পর্যন্ত।

এর পরে, α বলতে এখানে বোায় লার্নিং রেট কিংবা স্টেপ সাইজ। α -এর মান যত বড়ো হবে আমাদের লিনিয়ার রিপ্রেশন অ্যালগরিদম তত বড়ো বড়ো মান যোগ/বিয়োগ করে w_0, w_1 -এর মান আপডেট করবে। এর মান ঠিকমতো নেওয়ার বিশাল গুরুত্ব আছে। যদি α -এর মান অনেক বেশি বড়ো নিই, তাহলে অ্যালগরিদম আমাদের $J(w_0, w_1)$ -এর সর্বনিম্ন মানের দিকে আগমন অনেক দ্রুত, কিন্তু বড়ো বড়ো মান যোগ/বিয়োগ করে আপডেট করার কারণে (বড়ো বড়ো স্টেপ নেওয়ার কারণে) অনেক সময় আমরা $J(w_0, w_1)$ পয়েন্টটি পার হয়ে যেতে পারি (গ্রে

অধ্যায় ৩ : লিনিয়ার রিজেশন (Linear Regression)

৩.৪.১-এর বাঁয়ের ছবি)। আবার যদি α -এর মান অনেক বেশি তোটো নিই, তাহলে অ্যালগরিদম কনভার্জ (Converge) করতে অনেক বেশি সময় নেবে। অর্থাৎ সর্বনিম্ন $J(w_0, w_1)$ -তে পৌছাতে অনেক বেশি ইটারেশন লাগবে (গ্রাফ ৩.৪.১-এর ডানের ছবি)। তাই α -এর মান খুব সাবধানে পছন্দ করতে হবে, যেন বেশি বড়ো না হয়, ছোটোও না হয়। এটি সাধারণত 0.01 রাখা হয়ে থাকে।



ছবি ৩.৪.১

গ্রেডিয়েন্ট ডিসেন্টকে আরেকটু দ্রুততর করার জন্য এক ধরনের অপটিমাইজেশন চালানো হয়। এর পেছনে মূল ধারণাটি হচ্ছে, এটি গ্লোবাল মিনিমান থেকে যত দূরে থাকবে, লার্নিং রেট α -এর মান তত বড়ো থাকবে, সে যত গ্লোবাল মিনিমামের কাছাকাছি আসতে থাকবে, α -এর মান তত কমবে।

এখন, যেহেতু আপনি জানেন না যে গ্লোবাল মিনিমাম মানে আপনি আদৌ পৌছেছেন কি না, তাই এ ক্ষেত্রে একটি পদ্ধতি প্রয়োগ করা হয়। প্রতিবার লুপ চলার পরে, একবার করে error-এর মান পরীক্ষা করে দেখবেন। যদি দেখেন এরর আগের বারের চেয়ে কিছুটা কমেছে (10^{-10} -এর চেয়ে বেশি কমেছে), তাহলে α -এর মান 5% বাঢ়াবেন। অর্থাৎ α হয়ে যাবে 1.05α ।

আবার যদি দেখেন, এরর আগের চেয়ে তো কমেইনি বরং বেড়েছে, সে ক্ষেত্রে α -এর মান হিসেবে শেষবার আপডেট করার আগে α -এর মান যা ছিল তার 50% নেবেন, অর্থাৎ α হয়ে যাবে 0.5α । এই অপটিমাইজেশন পদ্ধতিটির নাম হচ্ছে **Bold Driver**। এরকম আরো কিছু অপটিমাইজেশন

আছে, যেমন – Adagrad, Adam, Adamax, RMSProp ইত্যাদি, যেগুলো আপনারা চাইলে
এই লিংক থেকে দেখে নিতে পারেন – <http://ruder.io/optimizing-gradient-descent/>

সবশেষে $\frac{\partial y}{\partial x}$ দিয়ে বোঝায় x -এর সাপেক্ষে y -এর পার্শ্বিয়াল ডেরিভেটিভ। আমরা আমাদের
সমীকরণে ব্যবহার করেছি $\frac{\partial}{\partial w_j} J(w_0, w_1)$ ।

$$\text{সুতরাং এখানে } x = w_j \text{ ও} \\ y = J(w_0, w_1)$$

অর্থাৎ আমরা j -এর প্রতিটি মানের জন্য মোট যতগুলো প্যারামিটার পাব (আমাদের ক্ষেত্রে 2টি,
 w_0 ও w_1) প্রতিটির সাপেক্ষে একবার করে $J(w_0, w_1)$ -এর পার্শ্বিয়াল ডেরিভেটিভ নেব।

সোজা করে বলতে গেলে, আমরা w_0 -এর সাপেক্ষে একবার $J(w_0, w_1)$ অর্থাৎ
 $\frac{1}{2m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}^2$ -কে ডিফারেন্সিয়েট করব, তারপরে w_1 -এর সাপেক্ষে আবার
 $J(w_0, w_1)$ অর্থাৎ $\frac{1}{2m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}^2$ -কে ডিফারেন্সিয়েট করব।

w_0 -এর সাপেক্ষে ডিফারেন্সিয়েট করে আমরা যা পাব, সেটি আমরা w_0 -এর মান আপডেট করার
সময় ব্যবহার করব। আর w_1 -এর সাপেক্ষে ডিফারেন্সিয়েট করে আমরা যা পাব তা w_1 -এর মান
আপডেট করার সময় ব্যবহার করব। প্রতিটি আলাদা প্যারামিটারের জন্য এভাবে আলাদা আলাদা
আপডেট করতে হবে।

এখন, আমরা যদি, w_0 -এর সাপেক্ষে একবার $\frac{1}{2m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}^2$ -কে
ডিফারেন্সিয়েট করি, তাহলে আমরা পাই –

$$\begin{aligned} \frac{\partial}{\partial w_0} J(w_0, w_1) &= \frac{\partial}{\partial w_0} \left[\frac{1}{2m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}^2 \right] \\ &= \frac{1}{m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\} \end{aligned}$$

একইভাবে, আমরা যদি, w_1 -এর সাপেক্ষে একবার $\frac{1}{2m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}^2$ -কে
ডিফারেন্সিয়েট করি, তাহলে আমরা পাই,

$$\begin{aligned} \frac{\partial}{\partial w_1} J(w_0, w_1) &= \frac{\partial}{\partial w_1} \left[\frac{1}{2m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}^2 \right] \\ &= \frac{1}{m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\} \cdot x^{(i)} \end{aligned}$$

অধ্যায় ৩ : লিনিয়ার রিগ্রেশন (Linear Regression)

যাঁদের মোটামুটি ক্যালকুলাসে হাত চালু আছে, তাঁরা চাইলে হিসাব করে মিলিয়ে দেখতে পারেন হিসাবটি এরকমই কি না। আমি সরাসরি উভয় দিয়ে দিলাম, যেহেতু আগেই বলেছি আমরা থিওরি আর কঠিন কঠিন অঙ্ক যতটুকু লাগে ঠিক ততটুকুই শিখব। যা-ই হোক, তাহলে আমরা w_0 ও w_1 এই দুটি প্যারামিটারের জন্য এই দুটি সমীকরণ ব্যবহার করব।

এখানে একটি বিষয় খেয়াল করে দেখবেন, প্রতিবার আমরা w_0 ও w_1 আপডেট করার সময় সমস্ত ট্রেনিং ডেটা ব্যবহার করছি। $\frac{1}{m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}$ থেকে দেখতে পাচ্ছি আমরা এখানে সমস্ত ট্রেনিং ডেটার ওপরে Error বের করে নিচ্ছি। এই ধরনের গ্রেডিয়েন্ট ডিসেন্টকে অনেক সময় Batch Gradient Descent বলে।

এর আরেকটি ভার্শন আছে, যেখানে সমস্ত ট্রেনিং ডেটা ব্যবহার করার বদলে শুধু যে ট্রেনিং ডেটার জন্য w_0 ও w_1 আপডেট করছি, শুধু সেই ট্রেনিং ডেটার জন্যই error বের করে ব্যবহার করে থাকি, অর্থাৎ শুধু $\{h_w(x^{(i)}) - y^{(i)}\}$ এতটুকু ব্যবহার করে থাকি, বাকি সব এক। সেটাকে বলা হয় Stochastic Gradient Descent। আপনারা যে-কোনোটিই ব্যবহার করতে পারেন, তবে আমি এই বইতে থিওরি বোঝানোর সময় ব্যাচ গ্রেডিয়েন্ট ডিসেন্টই ব্যবহার করব বেশিরভাগ সময়।

তাহলে এই হচ্ছে গ্রেডিয়েন্ট ডিসেন্ট অ্যালগরিদম ব্যবহার করে কীভাবে আমরা লিনিয়ার রিগ্রেশন করতে পারি, তার বিস্তারিত বর্ণনা। আশা করি সবাই বুঝতে পেরেছেন। যাঁদের এটি বুঝতে সমস্যা হচ্ছে এখনো, তাঁরা পরিচ্ছেদ ৩.৮-এ গ্রেডিয়েন্ট ডিসেন্ট ব্যবহার করে কীভাবে লিনিয়ার রিগ্রেশন করতে হয়, সে বর্ণনাটি দেখে নিতে পারেন পরবর্তী ৩.৫ পরিচ্ছেদে যাওয়ার আগেই।

পরিচ্ছেদ ৩.৫ : গ্রেডিয়েন্ট ডিসেন্ট (Gradient Descent) অ্যালগরিদম ব্যবহার করে মাল্টিভ্যারিয়েট (Multivariate) লিনিয়ার রিগ্রেশন

আমরা একক যে লিনিয়ার রিগ্রেশন করলাম, সেটি ছিল ইউনিভ্যারিয়েট (Univariate) লিনিয়ার রিগ্রেশন। এর অর্থ হচ্ছে, আমাদের ফিচার/ভ্যারিয়েবল ছিল মাত্র একটি (পিংজার সাইজ) যাকে আমরা শুধু x দিয়ে প্রকাশ করতাম এবং সেটি ব্যবহার করে আমরা পিংজার দাম আন্দজ করতাম।

এখন আমি যদি আরেকটি উদাহরণ নিই, যেখানে আমাদের একাধিক ফিচার আছে। যেরকম, আমরা যদি টেবিল ৩.৫.১ দেখি যেখানে কতগুলো ফোনের দাম দেওয়া আছে এবং সঙ্গে তার কিছু ফিচারের মান লেখা আছে (র্যাম, রম ইত্যাদি)।

রাম (GB)	রঘু (GB)	ক্যামেরা (MP)	ডিসপ্লে সাইজ (inch)	দাম (BDT)
2	16	8	5.0	10,000
4	32	8	5.2	12,500
6	64	12	5.5	15,000

টেবিল 3.5.1

এই টেবিল থেকে আমরা দেখতে পাচ্ছি, আমাদের ফিচারের সংখ্যা 4টি – রাম, রঘু, ক্যামেরা এবং ডিসপ্লে সাইজ। আর দাম হচ্ছে আমাদের আউটপুট। ফিচারগুলোকে আমরা এখন x_1, x_2, x_3 ও x_4 দিয়ে প্রকাশ করতে পারি। এদের জন্য Weight হবে তাহলে যথাক্রমে w_1, w_2, w_3 ও w_4 ।

এখন আমরা যদি এই একাধিক ফিচার ব্যবহার করে লিনিয়ার রিপ্রেশন করে দাম জানা নেওয়া এবং কোনো মোবাইলের ফিচারের মানগুলো আগের মতোই হাইপোথিসিসে বসিয়ে সেই মোবাইলের দাম বের করি, তাহলে সেই লিনিয়ার রিপ্রেশনকে বলা হবে মাল্টিভ্যারিয়েট (Multivariate) লিনিয়ার রিপ্রেশন।

এখন তাহলে মাল্টিভ্যারিয়েট লিনিয়ার রিপ্রেশনের হাইপোথিসিস সমীকরণটি লিখে ফেলা যাব। এটি আমাদের আগের ইউনিভ্যারিয়েট লিনিয়ার রিপ্রেশনের সঙ্গে অনেকখানি সামঞ্জস্যপূর্ণ:

$$h_w(x) = w_0 + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4$$

এখন আমরা যদি $x_0 = 1$ ধরি, তাহলে আমরা আমাদের সমীকরণ দাঁড়ায়,

$$h_w(x) = w_0x_0 + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4$$

একে জেনারালাইজ করে n -সংখ্যক ফিচারের জন্য লিখলে পাই,

$$h_w(x) = w_0x_0 + w_1x_1 + w_2x_2 + w_3x_3 + \dots + w_nx_n$$

আমরা এখন w নামে একটি ওয়েইট ভেক্টর (Weight Vector) চিন্তা করি, যেখানে $w = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \dots \\ w_n \end{bmatrix}$

অধ্যায় ৩ : লিনিয়ার রিগ্রেশন (Linear Regression)

এবং x নামে আরেকটি ফিচার ভেক্টর (Feature Vector) চিন্তা করি, যেখানে $x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}$

$$\text{এখন যদি আমরা } w^T x \text{ নিই, তাহলে আমরা পাই, } w^T x = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \dots \\ w_n \end{bmatrix}^T \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}$$

এখনে উল্লেখ্য, যে-কোনো ম্যাট্রিক্স A -এর ট্রান্সপোজ করলে ওই ম্যাট্রিক্স-এর রো (Row)-গুলো কলাম (Column) এবং কলামগুলো রো-তে রূপান্তরিত হয়ে যায় এবং একে A^T দিয়ে প্রকাশ করা হয়।

$$\begin{aligned} \text{সূতরাং, } w^T x &= \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \dots \\ w_n \end{bmatrix}^T \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix} \\ &= [w_0 \ w_1 \ w_2 \ \dots \ w_n] \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix} \end{aligned}$$

$$= w_0 x_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + \dots + w_n x_n \quad (\text{ম্যাট্রিক্স গুণনের নিয়ম অনুসারে})$$

সূতরাং, আমরা এখন লিখতে পারি যে,

$$\checkmark h_w(x) = w^T x$$

এটিই আমাদের মালিতভ্যারিয়েট লিনিয়ার রিগ্রেশনের হাইপোথিসিস।

আমাদের কস্ট ফাংশন এবং অবজেকটিভ ফাংশন আগের মতোই থাকবে।

কস্ট ফাংশনে সামান্য পরিবর্তন যেটি আসবে যে আমরা এখন w_n পর্যন্ত লিখব

$$J(w_0, w_1, w_2, \dots, w_n) = \frac{1}{2m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}^2$$

$$\text{আর, Objective Function} = \underset{w_0, w_1, w_2, \dots, w_n}{\text{minimize}} J(w_0, w_1, w_2, \dots, w_n)$$

$$= \underset{w_0, w_1, w_2, \dots, w_n}{\text{minimize}} \frac{1}{2m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}^2$$

তাহলে এখন আগের মতোই প্রেডিয়েন্ট ডিসেন্ট অ্যালগরিদম দিয়ে প্যারামিটার আপডেটের মাধ্যমে মাল্টিভ্যারিয়েট লিনিয়ার রিফ্রেশন চালাই, তাহলে আমাদের কাজের ধাপগুলো দাঁড়াচ্ছে -

- প্রথমে $w_0, w_1, w_2, \dots, w_n$ প্রত্যেককে র্যানডম মান অ্যাসাইন করি (আপনারা আগের মতোই সবগুলোর মান 1 দিয়ে শুরু করতে পারেন)
- এরপর $w_0, w_1, w_2, \dots, w_n$ -এর মান প্রতি ইটারেশনে একবার করে আপডেট করতে হবে এমনভাবে, যাতে $J(w_0, w_1, w_2, \dots, w_n)$ -এর মান কমে আসতে থাকে।
- আগের মতোই একটি সময় গিয়ে দেখা যাবে 5-10 ইটারেশনের পরেও $J(w_0, w_1, w_2, \dots, w_n)$ -এর মান আর উল্লেখযোগ্য হারে কমছে না। তখন আপনি আপনার লুপ চালানো শেষ করে দিতে পারেন।
- লুপ শেষ হয়ে গেলে আপনি $w_0, w_1, w_2, \dots, w_n$ -এর যে মানগুলো পেলেন, এটিই এদের অপটিমাল মান, যার জন্য আপনি $J(w_0, w_1, w_2, \dots, w_n)$ -এর মান সর্বনিম্ন পাচ্ছেন।
- এখন $w_0, w_1, w_2, \dots, w_n$ এদের মান হাইপোথিসিস $h_w(x) = w^T x$ -এ বসালেই আপনি আপনার কাজিক্ষিত হাইপোথিসিস পেয়ে যাবেন। এখন এতে কোনো অজানা $x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}$ -এর মান বসালেই আমরা তার জন্য কাজিক্ষিত আউটপুট কী হবে তা পেয়ে যাব।

সবশেষে যেটি বাকি থাকে, সেটি হচ্ছে আমরা $w_0, w_1, w_2, \dots, w_n$ এই প্যারামিটারগুলোর মান আপডেট কীভাবে করব? আমরা সেই আগের অ্যালগরিদমই অনুসরণ করব -

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(w_0, w_1, w_2, \dots, w_n)$$

আমাদেরকে $\frac{\partial}{\partial w_j} J(w_0, w_1, w_2, \dots, w_n)$ মান সেই আগের নিয়মেই বের করতে হবে। এর জন্য একটি সাধারণ সূত্র আছে -

$$\frac{\partial}{\partial w_j} J(w_0, w_1, w_2, \dots, w_n) = \frac{1}{m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\} \cdot x_j^{(i)}$$

অধ্যায় ৩ : লিনিয়ার রিজেশন (Linear Regression)

এখানে x_j দ্বারা বোকায় ট্রেনিং ডেটার j -তম ফিচার। সুতরাং $x_j^{(i)}$ দ্বারা বোকানো হচ্ছে i -তম ট্রেনিং ডেটার j -তম ফিচার। আরো ভেঙে বলি, টেবিল 3.5.1 থেকে দেখুন, এখানে $x^{(1)}$ মানে হচ্ছে টেবিলের প্রথম রো, যেটি প্রথম ট্রেনিং ডেটা নির্দেশ করছে। একইভাবে $x^{(2)}$ মানে টেবিলের দ্বিতীয় রো, যেটি দ্বিতীয় ট্রেনিং ডেটা নির্দেশ করছে।

আর $x_1^{(1)}$ মানে হচ্ছে টেবিলের প্রথম ট্রেনিং ডেটার প্রথম ফিচারের মান, অর্থাৎ প্রথম রো-এর প্রথম কলাম-এর মান অর্থাৎ 2। একইভাবে $x_2^{(2)}$ মানে হচ্ছে দ্বিতীয় ট্রেনিং ডেটার দ্বিতীয় ফিচারের ভ্যালু অর্থাৎ দ্বিতীয় রো-এর দ্বিতীয় কলাম-এর ভ্যালু অর্থাৎ 32।

এখন দেখুন, $j = 0$ এর জন্য,

$$\frac{\partial}{\partial w_0} J(w_0, w_1, w_2, \dots, w_n) = \frac{1}{m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\} \cdot x_0^{(i)}$$

আমরা হাইপোথিসিস দাঁড় করানোর সময় প্রথমেই ধরে নিয়েছিলাম, $x_0 = 1$

সুতরাং

$$\frac{\partial}{\partial w_0} J(w_0, w_1, w_2, \dots, w_n) = \frac{1}{m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\} \cdot 1$$

অর্থাৎ, $j = 0$ -এর জন্য

$$\frac{\partial}{\partial w_0} J(w_0, w_1, w_2, \dots, w_n) = \frac{1}{m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}$$

আর বাকি সব $j = 1, 2, 3, \dots, n$ -এর জন্য,

$$\frac{\partial}{\partial w_j} J(w_0, w_1, w_2, \dots, w_n) = \frac{1}{m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\} \cdot x_j^{(i)}$$

অর্থাৎ, $j = 1$ -এর জন্য,

$$\frac{\partial}{\partial w_1} J(w_0, w_1, w_2, \dots, w_n) = \frac{1}{m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\} \cdot x_1^{(i)}$$

আবার $j = 2$ -এর জন্য

$$\frac{\partial}{\partial w_2} J(w_0, w_1, w_2, \dots, w_n) = \frac{1}{m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\} \cdot x_2^{(i)}$$

এভাবে আমরা যতগুলো ফিচার নেব, প্রতিটির সাপেক্ষে একবার করে $J(w_0, w_1, w_2, \dots, w_n)$ -কে আমাদের পার্শিয়াল ডিফারেন্সিয়েশন করতে হবে এবং সেই অনুযায়ী প্যারামিটার আপডেট করতে হবে।

তাহলে এই হচ্ছে প্রেডিয়েন্ট ডিসেন্ট অ্যালগরিদম ব্যবহার করে কীভাবে আমরা মাল্টিভ্যারিয়েট লিনিয়ার রিগ্রেশন করতে পারি তার বিবরণ।

পরিচ্ছেদ ৩.৬ : ভেট্টর অ্যালজেব্রা (Vector Algebra) ব্যবহার করে লিনিয়ার রিগ্রেশন

এটি লিনিয়ার রিগ্রেশন বের করার সর্বশেষ পদ্ধতি যেটি আমরা দেখেব। এটি সরাসরি গাণিতিক একটি পদ্ধতি, এখানে প্রেডিয়েন্ট ডিসেন্টের মতো প্যারামিটার আপডেট করার কোনো ব্যাপার নেই। এটি বরং আমাদের প্রথম ব্যবহৃত নিউমেরিকাল অ্যানালাইসিস পদ্ধতিটির মতো একটি সরাসরি পদ্ধতি, যেখানে লিনিয়ার অ্যালজেব্রা ব্যবহৃত হয়েছে। পরিচ্ছেদ ৩.২-এ ব্যবহৃত সরাসরি পদ্ধতির সঙ্গে এই পদ্ধতির পার্থক্য হচ্ছে, পরিচ্ছেদ ৩.২-এর পদ্ধতিটি ব্যবহার করা যায় শুধু ইউনিভ্যারিয়েট লিনিয়ার রিগ্রেশনের ক্ষেত্রে, কারণ সেখানে শুধু একটি ফিচার বা ভ্যারিয়েবল নিয়ে কাজ করা হয়। কিন্তু ভেট্টর অ্যালজেব্রা ব্যবহার করে আমরা একাধিক ফিচারের জন্য অর্ধাং মাল্টিভ্যারিয়েট লিনিয়ার রিগ্রেশন-এর জন্য সরাসরি গাণিতিক পদ্ধতি ব্যবহার করে আমাদের সব প্যারামিটারের মান একেবারে বের করতে পারব।

আমরা আগের পরিচ্ছেদে ইতিমধ্যেই দেখেছি, মাল্টিভ্যারিয়েট লিনিয়ার রিগ্রেশনের ক্ষেত্রে আমাদের হাইপোথিসিস ছিল,

$$\checkmark h_w(x) = w^T x$$

এবং আমাদের কস্ট ফাংশন ছিল,

$$\checkmark J(w_0, w_1, w_2, \dots, w_n) = \frac{1}{2m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}^2$$

এখন আমরা যদি আমাদের সব ট্রেনিং ডেটাকে একটি ম্যাট্রিক্সের মধ্যে নিন্ত, যার প্রতিটি রো হচ্ছে একটি করে ট্রেনিং ডেটা আর প্রতিটি কলাম হচ্ছে একেকটি ফিচার, তাহলে ম্যাট্রিক্সটি হবে $m \times (n + 1)$ ডাইমেনশনের একটি ম্যাট্রিক্স, যেখানে মোট ট্রেনিং ডেটার সংখ্যা m , মোট ফিচারের সংখ্যা n ,

$$X = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & \dots & x_n^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & \dots & x_n^{(2)} \\ 1 & x_1^{(3)} & x_2^{(3)} & x_3^{(3)} & \dots & x_n^{(3)} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & x_1^{(m)} & x_2^{(m)} & x_3^{(m)} & \dots & x_n^{(m)} \end{bmatrix}$$

অধ্যায় ৩ : লিনিয়ার রিগ্রেশন (Linear Regression)

সেই সঙ্গে, যদি আমরা আরো একটি ম্যাট্রিক্স নিই, যেটি হবে একটি কলাম ম্যাট্রিক্স এবং তাতে m -সংখ্যক সারি থাকবে, যা আমাদের m -সংখ্যক টেনিং ডেটার প্রতিটির আউটপুটের আসল মানগুলো ধারণ করবে, তাহলে ম্যাট্রিক্সটি হবে $m \times 1$ ডাইমেনশনের একটি ম্যাট্রিক্স,

$$Y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

সবশেষে, আমাদের n -সংখ্যক ফিচারের জন্য সমস্ত ওয়েইট ($w_0, w_1, w_2, \dots, w_n$) নিয়ে একটি ওয়েইট ভেট্রে যদি চিন্তা করি, সেটি হবে $(n + 1) \times 1$ ডাইমেনশনের একটি ম্যাট্রিক্স,

$$w = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$$

এখন আমরা সরাসরি নিচের সূত্র প্রয়োগ করে প্রতিটি ওয়েইট ($w_0, w_1, w_2, \dots, w_n$)-এর অপটিমাল মান একবারেই পেয়ে যেতে পারি,

$$w = (x^T x)^{-1} x^T y$$

এখন থেকে w -এর মান পেয়ে গেলে সেটি আমাদের হাইপোথিসিস $h_w(x) = w^T x$ -এ বসিয়ে দিলেই আমরা পেয়ে যাব আমাদের কাঙ্ক্ষিত হাইপোথিসিসের সর্বশেষ সমীকরণ। এটিই হচ্ছে ভেট্রে অ্যালজেব্রা ব্যবহার করে এক বা একাধিক ফিচারের জন্য লিনিয়ার রিগ্রেশন করার পদ্ধতি।

এখন প্রশ্ন হচ্ছে কখন আমরা গ্রেডিয়েন্ট ডিসেন্ট ব্যবহার করব, আর কখন আমরা ভেট্রে অ্যালজেব্রা ব্যবহার করব? এর উত্তর হচ্ছে, যদি ফিচার সংখ্যা $n \leq 10,000$ হয়, তাহলে আমরা ভেট্রে অ্যালজেব্রা ব্যবহার করতে পারি, কেননা এতে কোনো লুপ বা ইটারেশন নেই এবং কোনো ফুরে নেওয়ার ব্যাপার নেই। তাই খুব দ্রুতই আমরা সমাধানে পৌঁছে যেতে পারি।

ওধু $n \leq 10,000$ -এর জন্যই কেন আমরা এটি ব্যবহার করতে পারি, এর পেছনে একটি কারণ হচ্ছে, w -এর সমীকরণটির দিকে তাকালে বুঝবেন, এখানে তিনটি ম্যাট্রিক্স মাল্টিপ্লিকেশন (Matrix Multiplication) অপারেশন করা হয়েছে এবং একটি ম্যাট্রিক্স ইনভার্শন (Matrix Inversion) অপারেশন করা হয়েছে। ধরে নিচ্ছি, সবাই-ই ম্যাট্রিক্স গুণন কীভাবে করতে হয় জানেন। আমি এখানে ওধু একটি 2×2 ম্যাট্রিক্স ইনভার্শনের সূত্রটি দিচ্ছি।

$$\text{ধরি, } A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$\text{তাহলে, } A^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

আশা করি দেখেই বুঝতে পারছেন, এটি বেশ জবরজৎ একটি পক্ষতি এবং যদি আমাদের n -এর মান অনেক বড়ো হয় (10,000-এর চেয়েও বড়ো, ধরি 11,000) তাহলে আমাদের –

X হবে $11,000 \times 11,000$ এই ডাইমেনশনের একটি ম্যাট্রিক্স।

X^T -ও হবে $11,000 \times 11,000$ এই ডাইমেনশনের একটি ম্যাট্রিক্স।

এখন এত বড়ো ডাইমেনশনের দুটি ম্যাট্রিক্সের গুণফল অবশ্যই Computationally Expensive হবে। এখানেই শেষ নয়। এই গুণফল $X^T X$ -এর ইনভার্স নিতে হবে, যেটি ম্যাট্রিক্স গুণফলের চেয়েও অনেক বেশি Computationally Expensive। সবশেষে, এই গুণফলের সঙ্গে আবার X^T গুণ করে তার সঙ্গে Y ম্যাট্রিক্সটি গুণ করলে আমরা সর্বশেষ ফলাফল পাব। এত বড়ো Computation কম্পিউটারের পক্ষে করা অনেক সময়সাপেক্ষ একটি ব্যাপার, তাই $n \leq 10,000$ পর্যন্ত আমরা এই ভেষ্টর অ্যালজেবরা মেথড ব্যবহার করব।

কিন্তু যদি ফিচার সংখ্যা $n > 10,000$ হয়, তাহলে প্রেডিয়েন্ট ডিসেন্ট ব্যবহার করতে হবে। কেননা, প্রেডিয়েন্ট ডিসেন্ট ফিচার সংখ্যা n -এর মান অনেক বড়ো হলেও ভালোভাবেই সমাধান পেঁচুতে পারে, যদিও এতে ইটারেশন অপারেশন আছে এবং সঠিকভাবে α বেছে নেওয়ার ব্যাপার আছে, যার কারণে এটি প্রায়ই ভেষ্টর অ্যালজেবরা মেথডের চেয়ে অনেক বেশি সময় ধরে চলে।

তাহলে এই হলো ভেষ্টর অ্যালজেবরা ব্যবহার করে কীভাবে আমরা লিনিয়ার রিপ্রেজেন্টেশন করতে পারি তার বিস্তারিত বর্ণনা।

পরিচ্ছেদ ৩.৭ : বায়াস (Bias), ভ্যারিয়েন্স (Variance) ও রেগুলারাইজেশন (Regularization)

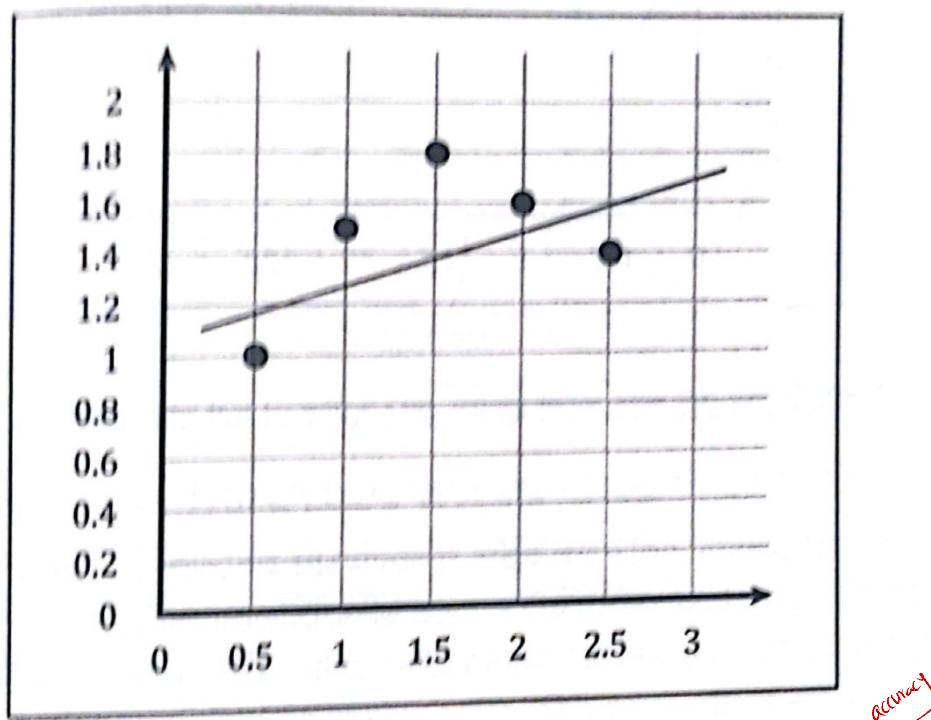
পরবর্তী সময়ে আরো কতগুলো ধারণা আমাদের প্রয়োজন হবে। সেগুলো আমি এখানেই আলোচনা করছি।

প্রথমেই আমরা বুঝে নিই **ওভারফিটিং (Overfitting)** ও **আন্ডারফিটিং (Underfitting)** কী। নাম শুনেই বোঝা যাচ্ছে যে, ওভারফিটিং মানে হচ্ছে বেশি হস্তিত্ব করে বেশি বেশি ফিট হয়ে গেছে, আবার আন্ডারফিটিং মানে হচ্ছে একেবারেই ফিট হয় নাই।

回归分析：线性回归分析 (Linear Regression)

ଆମେ ବିଷୟାତିକେ ଏହାଲେ ଚିନ୍ତା କରିବେ ପାରେମ । ଧରେ ନିମ ଆପଣି ଏକଟି ଶାର୍ଟ କିମ୍ବାଛି । ଦେଇ
କାହିଁ ଏହାରମାତ୍ରିକି ହସ୍ତା ଥାମେ ହାତେ ମେତି ପାରେ ସବେ, ଶୀତୋଶର ଶାର୍ଟଟି ଡାଙ୍ଗେର ନମେ ଏକେବାଳେ
ଝିଟୋପୀଟିକାରେ ଲେଖେ ଆଛେ । ଆପଣି ଯାଇ, ଯାମାନା ଏକଟି ମୋଡ଼ି ହସ୍ତ ଯାନ, ଦେଇ ଶାର୍ଟ ଆର ଆପଣାର
ପାଥେ ଘାଟିଲେ ମା । ଏକଇହାଲେ ଆମ୍ବାରମାତ୍ରିକି ମାନେ ହାତେ, ଆପଣି ଏମମ ଏକଟି ଶାର୍ଟ କିମ୍ବାଛି ଯେଉଁ
ଆପଣାର ପାଥେ ଏକେବାରେଇ ଫିଟିକି ହୁଅନି, ଏକମଧ୍ୟ ଡୋଲାଚାଲା ହସ୍ତ ଆଛେ ।

নিচের আইডি সেক্ষন (আই 3.7.1)। এই শ্রাফটিকে, আমরা যে সরলরেখা হাইপোথিসিস হিসেবে নিয়েছি, সেটি আমাদের ট্রেনিং ডেটার একেবারেই ফিট করতে না, অচের পরের না কর্তৃ আছে। এই ধরনের ফেরে নলা হয়ে যে, আমাদের হাইপোথিসিস আন্তর্বর্কিট হয়েছে, অর্থাৎ একেবারেই ফিটি হয়নি। ডেটা আছে একরকম, আর হাইপোথিসিস হয়েছে অন্যরকম।

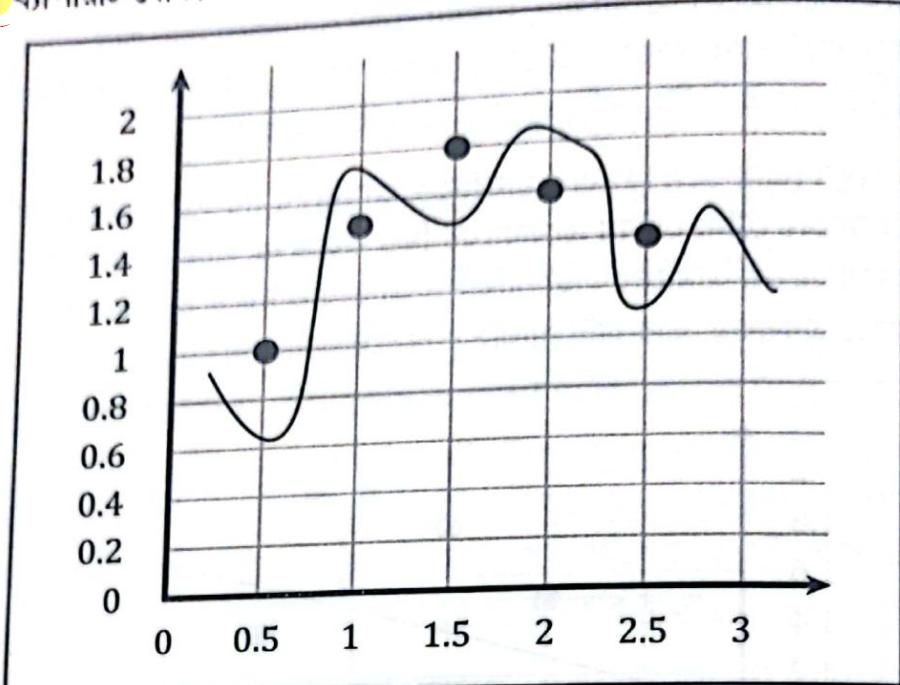


附录 3.7.1

এই ধরনের আন্ডারফিট হাইপোথিসিসের ফের্ডে বলা হয় যে এদের **হাই বায়াস (High Bias)** আছে, অর্থাৎ, হাইপোথিসিসগুলো আগে থেকেই ধারণা করে নিয়েছে যে ডেটা কীরকম হবে (আমাদের এই ফের্ডে সরলরেখিক) এবং সেই ধারণা ট্রেনিং ডেটা পর্যবেক্ষণ করার পরেও পরিবর্তিত হবে না। হাইপোথিসিস তাঁর নিজের ধারণার প্রতিই পক্ষপাতী থাকে। এটি অনেকটা জেড করে গো ধরে বসে থাকা ছোটো বাচ্চাদের মতো, যতই বোঝানো হোক (ট্রেনিং দেওয়া হোক) সে কথা শুনলে না (নিজের আকৃতি বদলাবে না)।

একইভাবে, উভারফিটিং বলতে বোাবা যে, আমাদের হাইপোথিসিস খুবই জটিল, প্রাচানো একটি হাইপোথিসিস এবং সেটি অধি ম্যানিং ডেটাতেই শুরু ভালোভাবে ফিট করে। ম্যানিং ডেটার বাইরে

যখন অন্য নতুন কোনো ডেটা আসে, সেই ডেটাকে আমাদের এই হাইপোথিসিস ফিট করাতে
পারবে না। ব্যাপারটি বোঝার জন্য নিচের গ্রাফ দেখুন (গ্রাফ 3.7.2)।



গ্রাফ 3.7.2

গ্রাফে হাইপোথিসিসের যেই কার্ড আঁকা হয়েছে, তার সমীকরণ ধরা যাক এরকম –

$$h_w(x) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1 x_2 + w_3 x_1^2 + w_4 x_2^3$$

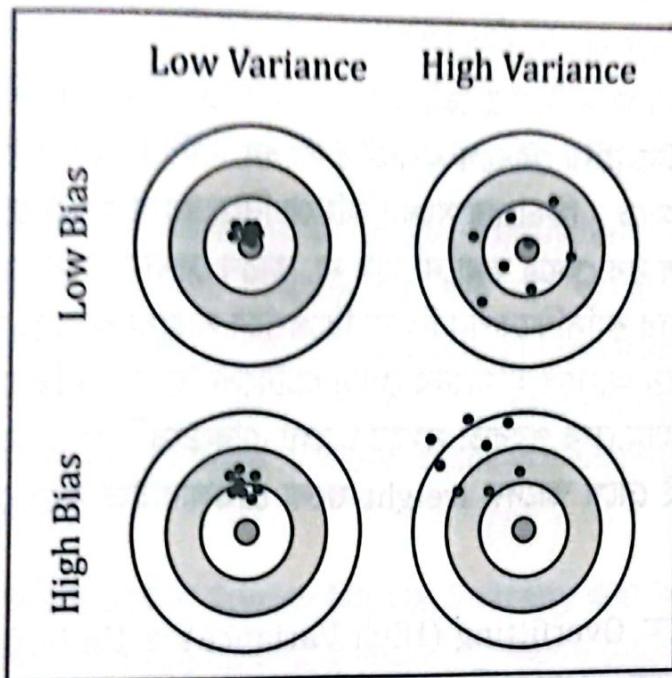
এখন দেখুন, কী বিদঘৃটে একটি হাই-অর্ডার পলিনোমিয়াল হাইপোথিসিস (প্রধান চলক x -এর সর্বোচ্চ ঘাত/পাওয়ার 1-এর বেশি), তাই না? এত কঠিন, ঘোরানো-প্যাঁচানো হাইপোথিসিস কেন ব্যবহার করা হয়েছে – শুধু যেন তা আমাদের ট্রেনিং ডেটাকে ঠিকভাবে ফিট করে, একটি ট্রেনিং পয়েন্টও যেন বাদ না যায়, অর্থাৎ, ট্রেনিং ডেটার সাপেক্ষে হাইপোথিসিসের কস্ট যেন শূন্য হয়।

এই ধরনের হাইপোথিসিস ট্রেনিং ডেটাতে খুব ভালোভাবে ফিট করবে অবশ্যই, কিন্তু সমস্যা হবে যখন ট্রেনিং ডেটার বাইরে অন্য কোনো ডেটা আসবে। তখন আর আমাদের এই প্যাঁচানো হাইপোথিসিস ভালোভাবে কাজ করবে না। এই ধরনের হাইপোথিসিস আসলে ট্রেনিং ডেটাতে এত ভালোভাবে ওভারফিট করে যে, এর বাইরে অন্য কোনো ডেটার জন্য সে নিজেকে জেনারালাইজ করতে পারে না আর; একেই বলে ওভারফিটিং। বলা হয় যে, Overfitted হাইপোথিসিসের High Variance থাকে। একটি হাইপোথিসিস দ্বারা ডিফাইন করা মডেল আসলে যত বেশি জটিল হয়, তত তার High Variance থাকে এবং তত তার Overfitted হওয়ার সম্ভাবনা থাকে।

consistency

অধ্যায় ৩ : লিনিয়ার রিজেশন (Linear Regression)

Bias-Variance Tradeoff বলে একটি টার্ম আছে, যেটি আপনারা কোথাও কোথাও হয়তো দেখবেন। আমাদের লক্ষ্য সব সময় থাকে আমরা Low Bias, Low Variance – এরকম কোনো একটি মডেল দাঁড় করাব। কিন্তু, বেশিরভাগ সময়েই আমরা এমন মডেল দাঁড় করাই, যেটাতে High Variance, High Bias কিংবা, High Variance, Low Bias অথবা Low Variance, High Bias থাকে।



ছবি 3.7.1

ওপরের ছবিটি (ছবি 3.7.1) দেখলে হয়তো আপনাদের আরেকটু পরিষ্কার ধারণা হবে ব্যাপারটা সম্পর্কে। অনেকেই হয়তো ডার্ট ছোড়ার খেলাটা খেলেছেন। আপনাদের হাতে কতগুলো ডার্ট থাকে, সেটি দিয়ে মাঝখানের ওই লাল বৃত্তটির মধ্যে ছুড়ে বিধাতে হয়। আমাদের এই ছবি 3.7.1-এ চারটি কেস দেওয়া আছে দেখবেন।

আমাদের লক্ষ্য থাকে সব সময় Low Bias (Accuracy), Low Variance (Consistency) পাওয়া, অর্থাৎ আমি যে ডার্ট ছুড়ব সেটি যেন মাঝখানের ওই লাল ছোটো বৃত্তে বেঁধে (Accuracy) এবং আমি যদি পরপর 10টি ডার্ট ছুড়ি, তাহলে 10টিই যেন লাল বৃত্তে গিয়ে বেঁধে (Consistency)। কিন্তু বেশিরভাগ সময়েই আমাদের হয়তো এটি achieve করা সম্ভব হয় না, প্রায় অসম্ভবই বলা চলে ব্যাপারটা। হয়তো, Accuracy ভালো হলে মডেল Inconsistent হয়, আবার Consistency থাকলেও মডেল Accurate হয় না। আবার কখনো কখনো কোনোটাই থাকে না।

এর কারণ হচ্ছে, সহজ করে বললে, একটি মডেলের Low Variance থাকা মানে মডেলটা Less Complex। সুতরাং, আমাদেরকে Low Variance যদি achieve করতে হয়, তাহলে Less

Complex কোনো একটি মডেল দিয়ে কাজ করতে হবে। আবার যদি আমরা Low Bias^{চাই} তাহলে আমাদেরকে Highly Complex মডেল ব্যবহার করতে হবে। এখন দেখুন, আমরা যদি দুটোই একসঙ্গে অর্জন করতে চাই, তাহলে আমাদের এমন একটি মডেল বেছে নিতে হবে, যেটি একই সঙ্গে Highly Complex আবার খুবই Low Complex, যেটি কখনোই সন্তুষ্ট নয়, তা না? তাই আসলে আমাদেরকে মাঝামাঝি কোনো একটি পয়েন্ট খুজে বের করতে হয়, এটাই Bias.

Variance Tradeoff!

Overfitting প্রধানত হয় যখন ফিচার ডেটার সংখ্যা আমাদের ট্রেনিং ডেটার চেয়ে অনেক বেশি হয়ে যাবে তখন (ডেটাসেটের টেবিলে কলামের সংখ্যা, সারির সংখ্যার চেয়ে বেশি হয়)। এর কমানোর দুটো উপায় আছে – ফিচারের সংখ্যা কমিয়ে নিয়ে আসা অথবা রেগুলারাইজেশন করা। ফিচারের সংখ্যা আসলে সব ক্ষেত্রে কমানো যায় না, কারণ দেখা যায় সবগুলো ফিচারই কোনো-না-কোনোভাবে আমাদের ক্লাসিফিকেশন কিংবা রিপ্রেশনে সাহায্য করছে। সুতরাং, একটি ফিচার ভ্যালু কমিয়ে ফেলা মানে আসলে আমাদের গোটা ডেটাসেটের একটি বিশাল পরিমাণ বাদ দিয়ে দেওয়া, যেটি আসলে আমাদের পরবর্তী সময়ে ভালো একিউরেসি পাওয়ার অন্তরক হতে পারে। তাই, সবগুলো ফিচারই রেখে তাদের weight-গুলো রেগুলারাইজ করে নেওয়াটাই বেশিরভাগ ক্ষেত্রে উভয় পদ্ধা।

এতক্ষণ আমরা দেখলাম Overfitting (High Variance) ও Underfitting (High Bias) বলতে আসলে কী বোঝায়। এখন আসি রেগুলারাইজেশনে।

রেগুলারাইজেশন মানে হচ্ছে সোজা বাংলায় হচ্ছে কোনো কিছুর গুরুত্বকে বা প্রভাবকে অন্য কোনো প্রভাবকের সাহায্যে কমানো-বাড়ানো। একে ফ্যানের রেগুলেটরের সঙ্গে তুলনা করা যেতে পারে। আমরা যখন ফ্যানের স্পিড কমানোর চেষ্টা করি, তখন আমরা কী করি? আমরা কি ফ্যানের পাখা ধরে ঝুলে পড়ি স্পিড কমানোর জন্য? কিংবা যদি আমরা ফ্যানের স্পিড বাড়াতে চাই তাহলে কী করি? ফ্যানের পাখা ধরে বাই বাই করে ঘোরানো শুরু করি কি? মোটেও না? আমরা ব্যবহার করি রেগুলেটর নামে ছোট একটি যন্ত্র, যেটি ঘূরিয়ে ঘূরিয়ে আমরা ফ্যানের স্পিড কমাই-বাড়াই।

রেগুলারাইজেশন ভালো কাজ করে সেইসব ক্ষেত্রে, যখন আমাদের বেশ অনেকসংখ্যক ফিচার ডেটা আছে এবং কোনো ফিচার ডেটাই আমাদের বাদ দেওয়ার উপায় নেই, সেই সব ক্ষেত্রে প্রতিটি ফিচারের সঙ্গে Associated থেকে Weight-গুলোকে পরিবর্তিত করে এইসব ফিচার ভ্যালুগুলোর গুরুত্ব কমানো-বাড়ানো যায়।

যেরকম, একটি উদাহরণ দিই। ধরা যাক, নিচের সমীকরণটি –

$$A \cdot B = 1$$

অধ্যায় ৩ : লিনিয়ার রিজেশন (Linear Regression)

এই সমীকরণে, যদি, A ও B দুটিকেই 1 ধরে নিই, তাহলে সমীকরণের উভয় পক্ষ সমান পাকে।
এখন যদি আমি চাই, আমি এই সমীকরণে B-এর প্রভাব কমাব, তাহলে কী করব?

একটু খেয়াল করে দেখুন, আমরা যদি আস্তে আস্তে A-এর মান বাড়াতে থাকি 1 থেকে 10, 100, 1000, 10000, 100000 এভাবে, তাহলে সমীকরণের উভয় পাশ সমান রাখার জন্য B-এর মান কিন্তু কমতে থাকবে 1 থেকে 0.1, 0.01, 0.001, 0.0001, 0.00001 এইভাবে। এই ঘটনাটি রেগুলারাইজেশন বোঝার জন্য একটি সাধারণ উদাহরণ। এটি দেওয়া হলো শুধু ব্যাপারটি সম্পর্কে একটু ধারণা পাওয়ার জন্য। আরেকটি জিনিস, যেহেতু B-কে কমানো-বাড়ানো আমরা A-এর মান বাড়ানো-কমানোর মাধ্যমে করছি, তাই A-কে বলা হয় Regularization Parameter।

আমরা রেগুলারাইজেশন মূলত ব্যবহার করি ওভারফিটিং কমানোর জন্য। আর ওভারফিটিং মূলত হয় যদি আমাদের মডেল খুব বেশি কমপ্লেক্স হয়, তাই না? আমরা মোটামুটি দুই ধরনের রেগুলারাইজেশনের সম্পর্কে জানব – একটি হচ্ছে L1 Regularization, আরেকটি হচ্ছে L2 Regularization। এদের আবার যথাক্রমে LASSO Regression ও RIDGE Regression-ও বলা হয়। কেন বলা হয়, এগুলো আপাতত আমাদের বইয়ের এখতিয়ারের বাইরে, এগুলো আরো কিছুটা অ্যাডভান্সড ধারণা।

এ দুটোর ধারণা বেশ জটিল, সেজন্য Sparse Matrix, Sparsity এবং আরো কিছু ধারণা জানা প্রয়োজন যেটি আসলে আপাতত এই হাতেখড়ির সময়ে না জানলেও চলবে বলে আমি মনে করি। সুতরাং, আপাতত সেগুলো বাদ রাখছি। আমাদের শুধু আপাতত এতটুকু জানলেই চলবে যে, যে-কোনো ধরনের রেগুলারাইজেশনের ক্ষেত্রেই আমরা আমাদের Weight Vector-এর মানগুলোকে কমিয়ে আনার চেষ্টা করি যতটুকু পারা যায়। এর একটি গালভরা নাম আছে – Penalizing the Weight Vectors।

L1 Regularization-এর ক্ষেত্রে আমরা যেটি রেগুলারাইজেশনের জন্য ব্যবহার করি, তা হলো –

$$\checkmark \frac{\lambda}{2m} \sum_{j=1}^n |w_j|$$

আর L2 Regularization-এর ক্ষেত্রে আমরা যেটি ব্যবহার করি সেটি হলো –

$$\checkmark \frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$

উভয় ক্ষেত্রেই ১ হচ্ছে রেগুলারাইজেশন প্যারামিটার, যেটি কিনা Weight Vector কমানোর সঙ্গে সমানুপাতিক। ১-এর মান যত হবে, যে-কোনো মডেলের তত বেশি কমপ্লেক্স থেকে সহজ হবার প্রবণতা বাড়বে। ১-এর মান শুন্য হলে কোনো রেগুলারাইজেশন হবে না।

একটি ছোটো উদাহরণ দিই। একটু আগেই আমরা ওভারফিটিং পড়ার সময় একটি বিচ্ছিন্ন প্র্যাচানো ও জটিল হাইপোথিসিস দেখলাম, যেটি ছিল এরকম –

$$h_w(x) = w_0 + w_1x_1 + w_2x_2 + w_3x_1x_2 + w_3x_1^2 + w_4x_2^3$$

এখন, দেখে কিছুটা হয়তো আপনারা আইডিয়া করতে পারছেন, যে এই হাইপোথিসিসে ওভারফিটিং হওয়ার সন্তান বেশ ভালো।

এখন, যদি আমরা ওভারফিটিং থেকে বাঁচতে চাই, তাহলে আমাদের একমাত্র ভবসা রেগুলারাইজেশন।

যেহেতু আমরা লিনিয়ার রিপ্রেশন করছি, এর জন্য কস্ট ফাংশন কীরকম হবে সেটি আমরা আগেই দেখেছি –

$$\frac{1}{2m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}^2$$

এখন, রেগুলারাইজেশন করা জন্য আমরা যেটি করব যে, আমাদের কস্ট ফাংশনের সঙ্গে একটি রেগুলাইজেশন টার্ম যুক্ত করে দেব ঠিক এভাবে –

$$\frac{1}{2m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}^2 + \frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$

এখন আমরা জানি যে, আমাদের অবজেকটিভ ফাংশন হচ্ছে এই কস্ট ফাংশনটিকে minimize করা। লক্ষ করুন, আমাদের ওপরের কস্ট ফাংশনটিতে দুটি টার্ম আছে, একটি সাধারণ Squared Error টার্ম, আরেকট হচ্ছে রেগুলারাইজেশন টার্ম।

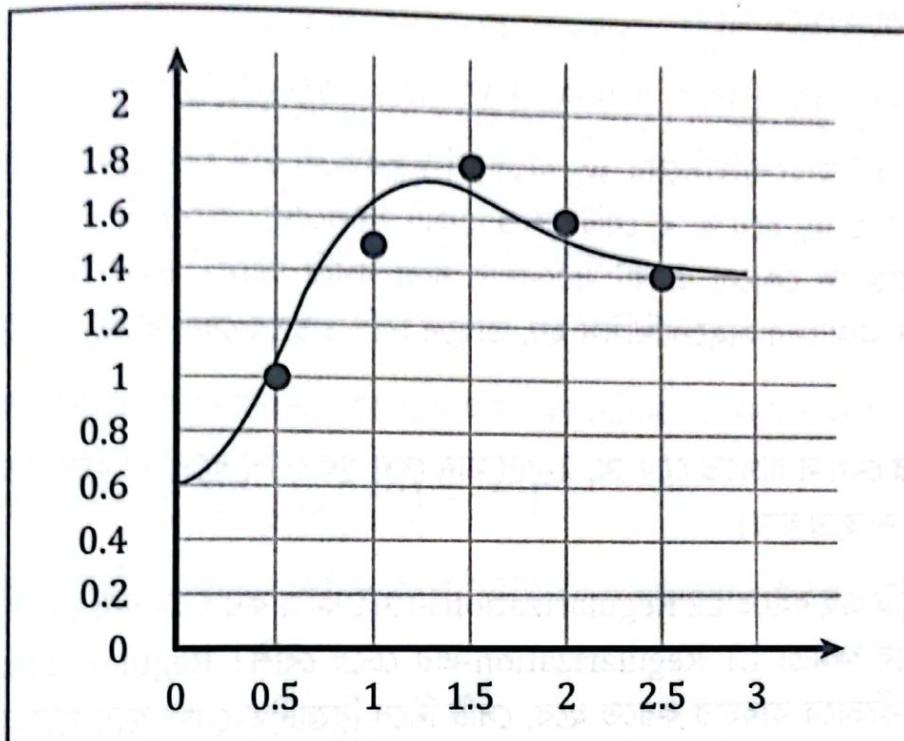
এখন, একটু ভালোমতো চিন্তা করে দেখুন তো, যেহেতু আমাদের সক্ষ্য হচ্ছে কস্ট ফাংশন মিনিমাইজ করা এবং আমাদের প্রথম টার্ম সাধারণভাবেই মিনিমাইজ হচ্ছে আগের মতোই সেটাতে কোনো সমস্যা নেই; কিন্তু যদি আমরা রেগুলারাইজেশন টার্মে λ -এর বেশ বড়েসড়ে একটি মান দিয়ে দিই (ধরা যাক, $\lambda = 1000$), তাহলে, সেই বড়ো মানের প্রভাব কমানোর জন্য (যেহেতু আমরা মিনিমাইজ করছি) $\sum_{j=1}^n w_j^2$ -এর মানকে খুব ছোটো মানের দিকে (প্রায় শূন্যে কাছাকাছি) কমে আসতে হবে, তাই না?

অর্থাৎ λ -এর বড়ো মানের প্রভাব কস্ট ফাংশনে কমিয়ে আনার জন্য আমাদের Weight Vector এর প্রতিটি weight কিংবা parameter-কে একেবারে কমিয়ে শূন্যের কাছাকাছি চলে আসতে হবে।

এটিই হচ্ছে রেগুলারাইজেশনের ধারণা এবং একটু চিন্তা করে দেখুন, আমাদের weight vector গুলোর মান যত ছোটো হতে থাকবে, তত আমাদের সেই বিচ্ছিন্ন হাইপোথিসিসটি এরকম

অধ্যায় ৩ : লিনিয়ার রিফ্রেশন (Linear Regression)

আঁকাৰাঁকা কাৰ্ড আকাৰ থেকে কিছুটা সুন্দৰ হয়ে গ্ৰাফ 3.7.3-এৰ মতো একটি আকাৰ হয়তো (কাছাকাছি কিছু একটা) ধাৰণ কৱবে।



গ্ৰাফ 3.7.3

গ্ৰাফটা আগেৰ মতো আঁকাৰাঁকা থেকে এৱকম সুন্দৰ smooth হয়ে আসাৰ মানে হচ্ছে আমাদেৱ
মডেলেৰ complexity আগেৰ চেয়ে অনেকাংশেই কমে এসেছে। অৰ্থাৎ, আমৱা আমাদেৱ
মডেলে ওভাৱফিটিং হওয়াৰ সন্ভাবনা অনেকটাই কমিয়ে এনেছি।

এখানে লক্ষণীয় যে, আমৱা কিন্তু সৱাসৱি weight vector-গুলোৰ মান কমাইনি, বৱং λ -এৰ মান
এমনভাৱে নিয়েছি, যাতে weight vector-গুলো কমে আসতে বাধ্য হয়, অৰ্থাৎ একটিৰ মান
বাড়িয়ে আমৱা আৱেকটিৰ মান কমিয়েছি, যেটি ছিল আমাদেৱ ৱেগুলারাইজেশনেৰ মূল ধাৰণা।

এখানে আৱেকটি ব্যাপার একটু খেয়াল রাখতে হবে, আমৱা ৱেগুলারাইজেশন কৱাৰ সময়
ৱেগুলারাইজেশন টাৰ্ম হিসেবে ব্যবহাৰ কৱছি—

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

এখন, একটু লক্ষ কৱবেন যে আমৱা কিন্তু এখানে $j = 1$ থেকে n পৰ্যন্ত weight vector-এৰ মান
নিয়েছি, একেবাৱে প্ৰথম যে weight vector ছিল আমাদেৱ w_0 , তাকে কিন্তু আমৱা ৱেগুলারাইজ
কৱছি না।

এখন, যদি আমরা λ -এর মান খুব বেশি বড়ো নিয়ে ফেলি, তাহলে, w_1 থেকে w_n -এর পর্যন্ত সবগুলোর মান রেগুলারাইজ করে আমরা শূন্যের একদম কাছাকাছি নিয়ে এলেও w_0 -এর মান কিন্তু যা ছিল, তা-ই থেকে যাবে।

$$h_w(x) = w_0 + w_1x_1 + w_2x_2 + w_3x_1x_2 + w_3x_1^2 + w_4x_2^3$$

তখন ওপরের এই হাইপোথিসিসের, w_0 বাদে বাকি সবগুলো টার্ম শূন্যের কাছাকাছি চলে যাবে ফলে – $h_w(x) \approx w_0$ হয়ে যাবে, যেটি মূলত একটি সরলরেখা। অনেকটা $y = 4$, এই দরজে x -অক্ষের সমান্তরাল কোনো একটি সরলরেখার কথা বলছি আমি। এখন শুধু এরকম একটি সরলরেখা যদি আমাদের হাইপোথিসিস হয়, তাহলে নিশ্চিতভাবে সেটি আন্ডারফিটিং হয়ে যাবে তাই না?

তাই, আমাদের খেয়াল রাখতে হবে যে, λ -এর মান যেন খুব বেশি বড়ো না হয়ে যায়; আবার খুব বেশি ছোটোও না হয়ে যায়।

আমরা মোটামুটি এই বইতে L2 Regularization নিয়েই কাজ করব, কেননা সেটির ওভারফিটিং এড়িয়ে যাওয়ার ক্ষমতা L1 Regularization-এর চেয়ে বেশি। Regularization আমাদের অ্যালগরিদমে কীভাবে ব্যবহার করতে হবে, সেটি নিয়ে বিস্তারিত লেখা হবে পরিচ্ছেদ 8.8-এ সেটি পড়লে আশা করি রেগুলারাইজেশনের ব্যবহার সম্পর্কে ধারণা পরিষ্কার হবে আরো।

পরিচ্ছেদ 3.8 : গ্রেডিয়েন্ট ডিসেন্ট পদ্ধতির সাহায্যে লিনিয়ার রিপ্রেশনের উদাহরণ

আশা করি, ইতিমধ্যে আমাদের লিনিয়ার রিপ্রেশনের ওপরে বেশ ভালো একটি ধারণা হয়ে আসে। এখন, গ্রেডিয়েন্ট ডিসেন্ট ব্যবহার করে কীভাবে এই লিনিয়ার রিপ্রেশন করা যায়, সেটি এবং দেখা যাক।

X	Y
1	1
2	3
4	3
3	2
5	5

টেবিল 3.8.1

একইভাবে বাকি ডেটাপয়েন্টগুলোর জন্য আমরা যদি হিসাব করি, তাহলে দেখব সবগুলোর জন্যই হাইপোথিসিসের মান আসছে শূন্য। অর্থাৎ মডেল-এর প্রেডিকশন হচ্ছে শূন্য।

নিচের চার্টটি দেখি :

X	Y (Original Output)	$h_w(x)$ (Predicted Output)	Error $= \{h_w(x^{(i)}) - y^{(i)}\}$	$\{h_w(x^{(i)}) - y^{(i)}\}^2$
1	1	0	-1	1
2	3	0	-3	9
4	3	0	-3	9
3	2	0	-2	4
5	5	0	-5	25
Total			-14	48

টেবিল 3.8.2

সূতরাং, $w_0 = 0$ ও $w_1 = 0$, এই প্যারামিটারের জন্য আমাদের ডেটাসেটের Cost হবে,

$$\begin{aligned}
 J(w_0, w_1) &= \frac{1}{2m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}^2 \\
 &= \frac{1}{2 \times 5} (1 + 9 + 9 + 4 + 25) [m = 5, \text{মোট ডেটাপয়েন্টের সংখ্যা}] \\
 &= \frac{48}{10} \\
 &= 4.8
 \end{aligned}$$

4.8 কিন্তু Cost হিসেবে বেশ বড়ো একটি মান, কারণ আমাদের লক্ষ্য ছিল Cost-এর মানে শূন্যতে নামিয়ে নিয়ে আসব। এখন, আমরা যেহেতু কেবল প্রথম ডেটাপয়েন্ট দেখেছি এতক্ষণে সূতরাং প্রথম ডেটাপয়েন্ট দেখার পরে আমাদের একবার Weight-এর মান আপডেট করতে হবে।

আমরা ধরে নিই, আমাদের লার্নিং রেট $\alpha = 0.01$

সূত্র অনুসারে,

$$w_0 = 0 - 0.01 \times \frac{1}{5} \times (-14) = 0.028$$

$$w_1 = 0 - 0.01 \times \frac{1}{5} \times (-14) \times 1 = 0.028$$

অধ্যায় ৩ : লিনিয়ার রিজেশন (Linear Regression)

এখন আমরা, এই নতুন $w_0 = 0.028$ ও $w_1 = 0.028$ নিয়ে আমাদের দ্বিতীয় ডেটা পয়েন্ট (2, 3)-এর জন্য হিসাব করব।

$$h_w(x) = 0.028 + 0.028 \times x$$

X	Y (Original Output)	$h_w(x)$ (Predicted Output)	Error $= \{h_w(x^{(i)}) - y^{(i)}\}$	$\{h_w(x^{(i)}) - y^{(i)}\}^2$
1	1	0.056	-0.944	0.891
2	3	0.084	-2.916	8.503
4	3	0.14	-2.86	8.179
3	2	0.112	-1.888	3.564
5	5	0.168	-4.832	23.348
Total			-13.44	44.485

টেবিল 3.8.3

$$\begin{aligned} \text{Cost}, J(w_0, w_1) &= \frac{1}{2m} \sum_{i=1}^m \{h_w(x^{(i)}) - y^{(i)}\}^2 \\ &= \frac{1}{2 \times 5} \times 44.485 \\ &= 4.4485 \end{aligned}$$

দেখা যাচ্ছে যে আমাদের Cost কিন্তু আগের চেয়ে নেমে এসেছে, অর্থাৎ বোঝা যাচ্ছে যে আমাদের প্রেডিয়েন্ট ডিসেন্ট কাজ করছে।

আমরা এভাবে করে বাকি তিনটি ডেটা পয়েন্টের জন্যও হিসাব করব, দেখব আমাদের Cost-এর মান আরো কমবে।

এভাবে আমাদের পাঁচটি ডেটা পয়েন্টের জন্য হিসাবনিকাশ শেষ হলে আমরা বলব যে, আমাদের প্রথম রাউন্ড শেষ হলো, অর্থাৎ প্রথম Epoch শেষ হলো। এরপর আমরা এই পাঁচটি ডেটা পয়েন্ট দিয়ে মডেলকে আরো এক রাউন্ড ট্রেনিং দেব। দ্বিতীয় রাউন্ডের শুরুতে w_0 ও w_1 মান হবে প্রথম রাউন্ড শেষে শেষবারের মতো আপডেট করার পরে w_0 ও w_1 -এর মান যা ছিল সেটি। এভাবে আমরা আরো বেশ কয়েক রাউন্ড মডেলকে ট্রেইন করব, ততক্ষণ পর্যন্ত যতক্ষণ না আমরা দেখব যে আমাদের Cost-এর মান আর খুব বেশি নামানো যাচ্ছে না।

ধৰা যাক, পঞ্চম রাউন্ডের ($5 \times 5 = 25$ বার Weight আপডেট করার পরে) শেষে আমাদের মোট Cost-এর মান দাঁড়িয়েছে 0.35। এর পরের ষষ্ঠ রাউন্ডে (5টি ডেটাপয়েন্টের জন্য আরো 5

বার আপডেট করার পর) সেটি নেমে দাঢ়াল 0.349, তারপরে সঙ্গম রাউন্ডের শেষে দাঢ়াল হয়তো 0.345। অর্থাৎ বোঝাই যাচ্ছে, শেষ দুই রাউন্ডে সর্বমোট 10 বার আপডেট করার পরেও আমাদের Cost-এর মান আর খুব একটা কমছে না, এখানেই কস্ট-এর গ্রাফ অনেকখানি ফ্ল্যাট হয়ে গেছে। সুতরাং, আমরা আর যতই ট্রেনিং দিই না কেন, Cost-এর মান আর খুব একটা কমবে না। অর্থাৎ আমাদের ট্রেনিং আমরা এখানে শেষ করতে পারি।

ট্রেনিং শেষ করে, ধরা যাক, আমাদের $w_0 = 0.2308$ ও $w_1 = 0.7904$ ।

তাহলে আমাদের সর্বশেষ হাইপোথিসিস হবে -

$$h_w(x) = 0.2308 + 0.7904 \times x$$

এখন এই হাইপোথিসিসে আমরা x -এর যে-কোনো মান বসালেই প্রেতিক্ষণ পেয়ে যাব যে, y -এর মান কত হবে এবং সেটি প্রকৃত y -এর মানের খুব কাছাকাছিই হবে।

এই ছিল Gradient Descent ব্যবহার করে কীভাবে আমরা লিনিয়ার রিগ্রেশন করতে পারি তার একটি উদাহরণ।