linear regression model: $\boldsymbol{y_i} = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \epsilon_i$ find the best values for BETA (OLS): **O**rdinary **L**east **S**quares. core idea: find the line (or hyperplane in multiple dimensions) that is as close as possible to all the data points simultaneously. Error (Residuals): $e_i = y_i - \hat{y}_i$ space: visualize residuals as the vertical distance from each data point to the regression line. Sum of Squared Residuals (SSR): we square each residual (which makes them all positive) and then sum them up. $SSR = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$ OLS:The goal of OLS: find the values of $\beta_0, \beta_1, \ldots, \beta_p$ that make SSR value as small as possible.(Works)Direct calculation using the Normal Equation.(Props)Provides an exact, optimal solution. No parameters to tune.(Cons)Can be computationally expensive with many features. Solving for the Coefficients: The Normal Equation: By taking the derivative of the SSR function with respect to each β coefficient and setting it to zero, we can solve for the optimal values.$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$
Gradient Descent: **G**heck the slope **T**ake a small step **R**epeat The size of each "step" you take is controlled by a parameter called the learning rate. (Works)Iterative; it takes steps to minimize error.(Props)More efficient for very large datasets. Very versatile.(Cons)Requires choosing a learning rate. May not find the exact minimum. Simple Linear Regression:a special case of the general model you showed earlier where you only have **O**ne predictor variable ($p = 1$).$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$ Minimize the sum of squares of err: Ordinary Least Squares (OLS):$\min_{a,b} \sum_{i=1}^{n} (y_i - a - bx_i)^2$ The Slope:$\hat{\beta}_1$, $\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$ In short, the slope is a measure of how X and Y vary together, scaled by how much X varies by itself The Intercept:$\hat{\beta}_0$, $\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$
Statistical Inference:To figure out the distribution of our estimates, we must make an assumption about the distribution of the errors. This is the most important assumption in linear regression for inference:Assumption: $\epsilon_i \overset{\text{i.i.d.}}{\sim} N(0, \sigma^2)$
Multiple Linear Regression:using just one predictor variable, we use multiple (p) predictors to explain our response variable.$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + \epsilon_i$ This is the most important new concept. In simple regression, β_1 was just the slope. In multiple regression, each coefficient has a more nuanced meaning:β_j is the average change in y for a one-unit increase in x_j, while holding all other predictors constant.
Deriving the Normal Equation: finding the best coefficients $(\beta_0, \beta_1, \ldots, \beta_p)$ using the Ordinary Least Squares (OLS)
Minimizing the Sum of Squares:
Using Calculus to Find the Minimum:To find the set of β values that results in the lowest possible error
The Derivative (Gradient): Setting the Gradient to Zero:The minimum of a function occurs where its slope is zero. This single equation is the general solution for finding the OLS coefficients for any linear regression model, no matter how many predictors you have. This is what statistical software calculates for you under the hood. Matrix Notation: By setting this equal to $\mathbf{y} - \boldsymbol{\epsilon}$, you perfectly recreate the entire set of 'n' equations in one clean statement. This compact form is what allows us to easily derive and compute the Normal Equation solution: $\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$.
Geometric Interpretation of OLS:The vector of fitted (predicted) values is: $\hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}}$ The vector of residuals is:$\hat{\boldsymbol{\epsilon}} = \mathbf{y} - \hat{\mathbf{y}} = \mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}$ The residual vector $\hat{\boldsymbol{\epsilon}}$ is orthogonal to the column space of $\mathbf{X}$ (the space spanned by the predictors). $\mathbf{X}^T\hat{\boldsymbol{\epsilon}} = \mathbf{0}$
Proof: $\mathbf{X}^T\hat{\boldsymbol{\epsilon}} = \mathbf{X}^T(\mathbf{y} - \hat{\mathbf{y}}) = \mathbf{X}^T(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}) = \mathbf{X}^T\mathbf{y} - \mathbf{X}^T\mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{X}^T\mathbf{y} - \mathbf{X}^T\mathbf{X}[(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}]$ (Substitute LSE) $= \mathbf{X}^T\mathbf{y} - (\mathbf{X}^T\mathbf{X})(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} = \mathbf{X}^T\mathbf{y} - \mathbf{I}\mathbf{X}^T\mathbf{y} = \mathbf{X}^T\mathbf{y} - \mathbf{X}^T\mathbf{y} = \mathbf{0}$
The Projection (Hat) Matrix: The fitted values $\hat{\mathbf{y}}$ are the orthogonal projection of $\mathbf{y}$ onto the column space of $\mathbf{X}$. This projection is performed by the projection matrix $\mathbf{H}$, also known as the "hat matrix". Projection Matrix:$\mathbf{H} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$ We can write the fitted values and residuals using $\mathbf{H}$: $\hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{X}[(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}] = [\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T]\mathbf{y} = \mathbf{H}\mathbf{y}$$\hat{\boldsymbol{\epsilon}} = \mathbf{y} - \hat{\mathbf{y}} = \mathbf{y} - \mathbf{H}\mathbf{y} = (\mathbf{I} - \mathbf{H})\mathbf{y}$
Properties of H via SVD: Let the Singular Value Decomposition of $\mathbf{X}$ (where $n > p$) be: $\mathbf{X}_{n\times p} = \mathbf{U}_{n\times p}\mathbf{D}_{p\times p}\mathbf{V}^T_{p\times p}$ where $\mathbf{U}^T\mathbf{U} = \mathbf{I}_p$ and $\mathbf{V}^T\mathbf{V} = \mathbf{I}_p$. The hat matrix $\mathbf{H}$ can be simplified to: $\mathbf{H} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T = \cdots = \mathbf{U}\mathbf{U}^T$ This form reveals two key properties of any projection matrix:
Symmetric:$\mathbf{H}^T = (\mathbf{U}\mathbf{U}^T)^T = (\mathbf{U}^T)^T\mathbf{U}^T = \mathbf{U}\mathbf{U}^T = \mathbf{H}$
Idempotent:$\mathbf{H}^2 = (\mathbf{U}\mathbf{U}^T)(\mathbf{U}\mathbf{U}^T) = \mathbf{U}(\mathbf{U}^T\mathbf{U})\mathbf{U}^T = \mathbf{U}\mathbf{I}_p\mathbf{U}^T = \mathbf{U}\mathbf{U}^T = \mathbf{H}$
Eigenvalues of H: The eigenvalues λ of an idempotent matrix must be 0 or 1. Step1:Let $\mathbf{v} \neq \mathbf{0}$ be an eigenvector of $\mathbf{H}$ with eigenvalue λ. $\mathbf{H}\mathbf{v} = \lambda\mathbf{v}$
Apply $\mathbf{H}$ again: $\mathbf{H}^2\mathbf{v} = \mathbf{H}(\lambda\mathbf{v}) = \lambda(\mathbf{H}\mathbf{v}) = \lambda(\lambda\mathbf{v}) = \lambda^2\mathbf{v}$
Since: $\mathbf{H}$ is idempotent, $\mathbf{H}^2 = \mathbf{H}$, which means $\mathbf{H}^2\mathbf{v} = \mathbf{H}\mathbf{v}$.
Get: $\lambda^2\mathbf{v} = \lambda\mathbf{v} \implies (\lambda^2 - \lambda)\mathbf{v} = \mathbf{0}$ Since $\mathbf{v} \neq \mathbf{0}$, we must have $\lambda(\lambda - 1) = 0$, so $\lambda = 0$ or $\lambda = 1$.
Statistical Inference for Multiple Regression:
Model Assumptions (Matrix Form):
1: $E[\boldsymbol{\epsilon}] = \mathbf{0}$ 2: $Cov(\boldsymbol{\epsilon}) = \sigma^2\mathbf{I}_n$ (Homoscedasticity and uncorrelated errors) 3: For inference: $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \sigma^2\mathbf{I}_n)$
Distribution of LSE:
Under the normality assumption: the LSE $\hat{\boldsymbol{\beta}}$ follows a multivariate normal distribution: $\hat{\boldsymbol{\beta}} \sim N\left(\boldsymbol{\beta}, \sigma^2(\mathbf{X}^T\mathbf{X})^{-1}\right)$
This confirms the estimator is unbiased: as $E[\hat{\boldsymbol{\beta}}] = \boldsymbol{\beta}$. For an individual coefficient:$\hat{\beta}_j$: $\frac{\hat{\beta}_j - \beta_j}{\text{s.e.}(\hat{\beta}_j)} \sim t_{n-p-1}$ where $p+1$ is the number of parameters (including intercept).
Distribution of Residuals:The Residual Sum of Squares (RSS), also called SS_{Error}, follows a Chi-squared distribution $\frac{SS_{Error}}{\sigma^2} = \frac{||\mathbf{y} - \hat{\mathbf{y}}||^2}{\sigma^2} \sim \chi^2_{n-p-1}$ This provides the unbiased estimator for the error variance σ^2: $s^2 = \frac{SS_{Error}}{n-p-1}$
Confidence and Prediction Intervals:Let $t^* = t_{n-p-1}(\alpha/2)$ be the critical value from the t-distribution.
Confidence Interval for a Parameter β_j
A $100(1-\alpha)\%$ CI for β_j is: $\hat{\beta}_j \pm t^* \cdot \text{s.e.}(\hat{\beta}_j)$ where $\text{s.e.}(\hat{\beta}_j) = s\sqrt{c_{jj}}$ and c_{jj} is the j-th diagonal element of $(\mathbf{X}^T\mathbf{X})^{-1}$.
Confidence Interval for Mean Response: For a given input vector $\mathbf{x}_0$, the $100(1-\alpha)\%$ CI for the mean response $E[y_0] = \mathbf{x}_0^T\boldsymbol{\beta}$ is: $\hat{y}_0 \pm t^* \cdot s\sqrt{\mathbf{x}_0^T(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{x}_0}$ where $\hat{y}_0 = \mathbf{x}_0^T\hat{\boldsymbol{\beta}}$.
Prediction Interval for Individual Response: For a single future observation y_0 at $\mathbf{x}_0$, the $100(1-\alpha)\%$ PI is: $\hat{y}_0 \pm t^* \cdot s\sqrt{1 + \mathbf{x}_0^T(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{x}_0}$ The '1' under the square root accounts for the additional uncertainty of the individual error term ϵ_0.
Gauss-Markov Theorem:
Assumptions (Gauss-Markov Assumptions):

- $E[\boldsymbol{\epsilon}] = \mathbf{0}$ (Errors have zero mean)
- $Cov(\boldsymbol{\epsilon}) = E[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T] = \sigma^2\mathbf{I}_n$ (Errors are homoscedastic and uncorrelated)

definition[BLUE] An estimator is **B**est **L**inear **U**nbiased **E**stimator (BLUE) if:

- **Linear:** It is a linear function of the response $\mathbf{y}$ (e.g., $\hat{\boldsymbol{\beta}} = \mathbf{A}\mathbf{y}$).
- **Unbiased:** Its expected value is the true parameter ($E[\hat{\boldsymbol{\beta}}] = \boldsymbol{\beta}$).
- **Best:** It has the minimum variance among all linear unbiased estimators.

Gauss-Markov Under the Gauss-Markov assumptions, the Least Squares Estimator (LSE) $\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$ is BLUE.

- The LSE is linear by definition: $\hat{\boldsymbol{\beta}} = \mathbf{A}_{LSE}\mathbf{y}$ where $\mathbf{A}_{LSE} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$.
- The LSE is unbiased: $E[\hat{\boldsymbol{\beta}}] = E[\mathbf{A}_{LSE}(\mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon})] = \mathbf{A}_{LSE}\mathbf{X}\boldsymbol{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{X}\boldsymbol{\beta} = \mathbf{I}\boldsymbol{\beta} = \boldsymbol{\beta}$.
- Let $\tilde{\boldsymbol{\beta}} = \mathbf{A}\mathbf{y}$ be any other linear estimator.
- For $\tilde{\boldsymbol{\beta}}$ to be unbiased, $E[\mathbf{A}\mathbf{y}] = \mathbf{A}\mathbf{X}\boldsymbol{\beta} = \boldsymbol{\beta}$ must hold for all $\boldsymbol{\beta}$, which requires $\mathbf{A}\mathbf{X} = \mathbf{I}$.
- Let $\mathbf{A} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T + \mathbf{D}$ for some matrix $\mathbf{D}$.
- The unbiasedness condition $\mathbf{A}\mathbf{X} = \mathbf{I}$ implies: $[(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T + \mathbf{D}]\mathbf{X} = \mathbf{I} \implies (\mathbf{X}^T\mathbf{X})^{-1}(\mathbf{X}^T\mathbf{X}) + \mathbf{D}\mathbf{X} = \mathbf{I}\ \mathbf{I} + \mathbf{D}\mathbf{X} = \mathbf{I} \implies \mathbf{D}\mathbf{X} = \mathbf{0}$
- Now, find the variance of $\tilde{\boldsymbol{\beta}}$: $Var(\tilde{\boldsymbol{\beta}}) = Var(\mathbf{A}\mathbf{y}) = \mathbf{A}Var(\mathbf{y})\mathbf{A}^T = \mathbf{A}(\sigma^2\mathbf{I})\mathbf{A}^T = \sigma^2\mathbf{A}\mathbf{A}^T$
- Substitute $\mathbf{A} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T + \mathbf{D}$: $[(\mathbf{A}\mathbf{A}^T = [(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T + \mathbf{D}][(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T + \mathbf{D}]^T = [(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T + \mathbf{D}][\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1} + \mathbf{D}^T] = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1} + (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{D}^T + \mathbf{D}\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1} + \mathbf{D}\mathbf{D}^T)]$
- Since $\mathbf{D}\mathbf{X} = \mathbf{0}$, the third term is $\mathbf{0}$. Since $(\mathbf{D}\mathbf{X})^T = \mathbf{X}^T\mathbf{D}^T = \mathbf{0}$, the second term is also $\mathbf{0}$. $[(\mathbf{A}\mathbf{A}^T = (\mathbf{X}^T\mathbf{X})^{-1} + \mathbf{D}\mathbf{D}^T)]$
- Therefore, the variance of the other estimator is: $[(Var(\tilde{\boldsymbol{\beta}}) = \sigma^2[(\mathbf{X}^T\mathbf{X})^{-1} + \mathbf{D}\mathbf{D}^T] = \sigma^2(\mathbf{X}^T\mathbf{X})^{-1} + \sigma^2\mathbf{D}\mathbf{D}^T = Var(\hat{\boldsymbol{\beta}}_{LSE}) + \sigma^2\mathbf{D}\mathbf{D}^T)]$
- The matrix $\mathbf{D}\mathbf{D}^T$ is positive semi-definite. This means its diagonal elements (which correspond to the variances of the individual $\tilde{\beta}_j$) are non-negative.
- Thus, $Var(\tilde{\beta}_j) = Var(\hat{\beta}_{LSE,j}) +$ (a non-negative term). The variance of the LSE is minimal, making it the "Best" linear unbiased estimator.

Classification - Why Not Linear Regression?:
For a binary classification problem with $y \in \{0, 1\}$, a linear regression model would be: $Y \approx \beta_0 + \beta_1 X$
We could interpret the prediction $\hat{y}$ as a probability: $P(Y = 1|X) = \beta_0 + \beta_1 X$
This approach is flawed because the output of a linear model is not constrained to the interval [0, 1], and thus can produce invalid probabilities.
Logistic Regression: Instead of modeling the response directly, logistic regression models the probability $P(Y = 1|X)$. It uses the **logistic function** (or sigmoid function) to ensure the output is between 0 and 1. For a single predictor X: $P(Y = 1|X) = p(X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}$
For a multi-variable input vector $\mathbf{x}$: $P(Y = 1|\mathbf{x}) = p(\mathbf{x}) = \frac{e^{\beta^T\mathbf{x}}}{1+e^{\beta^T\mathbf{x}}} = \frac{1}{1+e^{-\beta^T\mathbf{x}}}$ The probability for class 0 is therefore: $P(Y = 0|\mathbf{x}) = 1 - p(\mathbf{x}) = \frac{1}{1+e^{\beta^T\mathbf{x}}}$
The Logit Transformation: The logistic model can be re-written to show a linear relationship with the **log-odds** (or **logit**).
Odds: The ratio of the probability of an event happening to the probability of it not happening. Odds $= \frac{p(\mathbf{x})}{1-p(\mathbf{x})} = \frac{e^{\beta^T\mathbf{x}}/(1+e^{\beta^T\mathbf{x}})}{1/(1+e^{\beta^T\mathbf{x}})} = e^{\beta^T\mathbf{x}}$
Log-Odds (Logit): Taking the natural logarithm of the odds reveals a linear model. $\log(\text{Odds}) = \log\left(\frac{p(\mathbf{x})}{1-p(\mathbf{x})}\right) = \beta^T\mathbf{x}$
This shows that the log-odds of the outcome are a linear function of the predictors. A one-unit increase in a predictor x_j changes the log-odds by β_j. Parameter Estimation: Maximum Likelihood (MLE):
The coefficients β are estimated using **Maximum Likelihood Estimation (MLE)**. The goal is to find the β that maximizes the probability of observing the given data.
Likelihood Function: For n independent observations $(\mathbf{x}_i, y_i)$, the likelihood function is the product of the individual probabilities. For a binary outcome $y_i \in \{0, 1\}$, the probability for a single observation i can be written compactly as: $P(y_i|\mathbf{x}_i) = p(\mathbf{x}_i)^{y_i}(1 - p(\mathbf{x}_i))^{1-y_i}$ The total likelihood function $L(\beta)$ is the product over all n observations: $L(\beta) = \prod_{i=1}^{n} P(y_i|\mathbf{x}_i) = \prod_{i=1}^{n} p(\mathbf{x}_i)^{y_i}(1 - p(\mathbf{x}_i))^{1-y_i}$
Log-Likelihood Function: It is mathematically easier to maximize the **log-likelihood function** $\ell(\beta)$, which turns the product into a sum: $\ell(\beta) = \log(L(\beta)) = \sum_{i=1}^{n}[y_i \log(p(\mathbf{x}_i)) + (1 - y_i)\log(1 - p(\mathbf{x}_i))]$ We can substitute the logistic function definitions: $[(\log(p(\mathbf{x}_i)) = \log\left(\frac{e^{\beta^T\mathbf{x}_i}}{1+e^{\beta^T\mathbf{x}_i}}\right) = \beta^T\mathbf{x}_i - \log(1 + e^{\beta^T\mathbf{x}_i})$
$\log(1 - p(\mathbf{x}_i)) = \log\left(\frac{1}{1+e^{\beta^T\mathbf{x}_i}}\right) =$
$- \log(1 + e^{\beta^T\mathbf{x}_i}))]$ Substituting these back into $\ell(\beta)$:
$[(\ell(\beta) = \sum_{i=1}^{n}\left[y_i(\beta^T\mathbf{x}_i - \log(1 + e^{\beta^T\mathbf{x}_i})) + (1 - y_i)(-\log(1 + e^{\beta^T\mathbf{x}_i}))\right] = \sum_{i=1}^{n}\left[y_i\beta^T\mathbf{x}_i - y_i\log(1 + e^{\beta^T\mathbf{x}_i}) - \log(1 + e^{\beta^T\mathbf{x}_i}) + y_i\log(1 + e^{\beta^T\mathbf{x}_i})\right] = \sum_{i=1}^{n}\left[y_i\beta^T\mathbf{x}_i - \log(1 + e^{\beta^T\mathbf{x}_i})\right])]$
There is no closed-form solution for $\hat{\beta}$ that maximizes this function. We must use an iterative numerical optimization algorithm.
Numerical Optimization: Newton-Raphson: The **Newton-Raphson** algorithm is used to find the roots of a function, i.e., where $f(x) = 0$. To maximize $\ell(\beta)$, we use it to find the roots of the first derivative (the gradient), i.e., where $\nabla\ell(\beta) = 0$.
General 1D Case: To find the root of $f(x)$, we start with a guess x^{old}. We approximate $f(x)$ using a first-order Taylor expansion around x^{old}:
$f(x) \approx f(x^{old}) + f'(x^{old})(x - x^{old})$
We seek $x = x^{new}$ such that $f(x^{new}) = 0$:
$0 \approx f(x^{old}) + f'(x^{old})(x^{new} - x^{old})$
Solving for x^{new} gives the update rule:
$x^{new} = x^{old} - \frac{f(x^{old})}{f'(x^{old})}$
Multivariate Case (for Logistic Regression): For a vector β, the algorithm (also known as Iteratively Reweighted Least Squares, or IRLS) becomes: $\beta^{new} = \beta^{old} - H^{-1}\nabla\ell(\beta^{old})$
Where: $\nabla\ell(\beta) = \frac{d\ell(\beta)}{d\beta}$ is the **gradient** (vector of first partial derivatives). $H = \frac{d^2\ell(\beta)}{d\beta d\beta^T}$ is the **Hessian** (matrix of second partial derivatives).
Decision Boundary: Once $\hat{\beta}$ is found, we classify a new observation $\mathbf{x}_0$ based on its predicted probability. The decision boundary is the set of points where the model is uncertain.
Decision Rule: Predict $y = 1$ if $\hat{P}(y = 1|\mathbf{x}_0) > \delta$, where δ is a threshold (commonly 0.5).
Odds Rule: This is equivalent to predicting $y = 1$ if the odds are greater than $\frac{\delta}{1-\delta}$. For $\delta = 0.5$, this threshold is 1. $\frac{\hat{P}(y=1|\mathbf{x}_0)}{1-\hat{P}(y=1|\mathbf{x}_0)} = \exp(\hat{\beta}^T\mathbf{x}_0) > 1$
Linear Boundary: Taking the log of both sides gives the decision boundary: $\hat{\beta}^T\mathbf{x}_0 > \log(1) \implies \hat{\beta}^T\mathbf{x}_0 > 0$
The decision boundary is the hyperplane defined by $\hat{\beta}^T\mathbf{x} = 0$. This is why logistic regression is a **linear classifier**.
Linear Discriminant Analysis (LDA): LDA is an alternative classification method that models the class-conditional densities $f_k(x) = f(X = x|Y = k)$ and uses Bayes' theorem.
Bayes' Theorem for Classification: The posterior probability that an observation x belongs to class k is: $p_k(x) = P(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K}\pi_l f_l(x)}$ Where:$p_k(x)$ is the **posterior probability**.$\pi_k = P(Y = k)$ is the **prior probability** of class k.$f_k(x)$ is the **class-conditional density** of X for an observation from class k.The classification rule is to assign x to the class k for which $p_k(x)$ is largest.
LDA Assumptions: $f_k(x)$ is assumed to be a p-dimensional multivariate **Gaussian (Normal)** distribution: $X|Y = k \sim \mathcal{N}(\mu_k, \Sigma_k)$. All classes share a **common covariance matrix**: $\Sigma_k = \Sigma$ for all k.The density function for class k is: $f_k(x) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}}\exp\left(-\frac{1}{2}(x - \mu_k)^T\Sigma^{-1}(x - \mu_k)\right)$
Derivation of the Linear Discriminant Function: To find the class k that maximizes $p_k(x)$, we only need to maximize the numerator $\pi_k f_k(x)$. Maximizing the log is equivalent and simpler: $\arg\max_k\left(\log(\pi_k) + \log(f_k(x))\right)$ Substitute the Gaussian density $f_k(x)$: $\log(f_k(x)) = \log\left(\frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}}\right) - \frac{1}{2}(x - \mu_k)^T\Sigma^{-1}(x - \mu_k)$ The term $\log\left(\frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}}\right)$ is constant for all k (due to the common Σ) and can be dropped. We are left with maximizing: $\arg\max_k\left(\log(\pi_k) - \frac{1}{2}(x - \mu_k)^T\Sigma^{-1}(x - \mu_k)\right)$
Expanding the quadratic term: $(x - \mu_k)^T\Sigma^{-1}(x - \mu_k) = x^T\Sigma^{-1}x - 2x^T\Sigma^{-1}\mu_k + \mu_k^T\Sigma^{-1}\mu_k$ The term $x^T\Sigma^{-1}x$ is also constant for all k and can be dropped. We are left with maximizing the **linear discriminant function**, $\delta_k(x)$: $\delta_k(x) = \log(\pi_k) - \frac{1}{2}\left(-2x^T\Sigma^{-1}\mu_k + \mu_k^T\Sigma^{-1}\mu_k\right)$
$\delta_k(x) = x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k + \log(\pi_k)$ This function is **linear** in x. The decision boundary between class k and l is the set of points where $\delta_k(x) = \delta_l(x)$, which defines a linear hyperplane.
Parameter Estimation for LDA:
In practice, the parameters π_k, μ_k, and Σ are unknown and must be estimated from the training data: $[(\hat{\pi}_k = \frac{n_k}{n}$ (proportion of samples in class k)
$\hat{\mu}_k = \frac{1}{n_k}\sum_{i:y_i=k} x_i$ (mean of samples in class k)
$\hat{\Sigma} = \frac{1}{n-K}\sum_{k=1}^{K}\sum_{i:y_i=k}(x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T$ (pooled covariance))]
Fisher's Interpretation: LDA can also be seen as a dimensionality reduction technique that finds the projection vector w that maximizes the ratio of between-class variance to within-class variance.
Within-Class Covariance:
$\hat{\Sigma}_W = \frac{1}{n-K}\sum_{k=1}^{K}\sum_{i:y_i=k}(x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^\top$
Between-Class Covariance: $\hat{\Sigma}_B = \sum_{k=1}^{K} n_k(\hat{\mu}_k - \hat{\mu})(\hat{\mu}_k - \hat{\mu})^\top$ **Objective (Rayleigh Quotient):** $\max_w \frac{w^\top\hat{\Sigma}_B w}{w^\top\hat{\Sigma}_W w}$
Quadratic Discriminant Analysis (QDA):QDA Assumption: QDA follows the same logic as LDA, but it drops the key assumption of a common covariance matrix. $X|Y = k \sim \mathcal{N}(\mu_k, \Sigma_k)$ Each class k has its **own covariance matrix** Σ_k.
Derivation of the Quadratic Discriminant Function: We start from the same log-maximization, but we can no longer drop the terms involving Σ_k: $\arg\max_k\left(\log(\pi_k) + \log(f_k(x))\right)$
$\arg\max_k\left(\log(\pi_k) + \log\left(\frac{1}{(2\pi)^{p/2}|\Sigma_k|^{1/2}}\right) - \frac{1}{2}(x - \mu_k)^T\Sigma_k^{-1}($
The term $\log(|\Sigma_k|)$ is now specific to class k and cannot be dropped. The term $x^T\Sigma_k^{-1}x$ from expanding the quadratic form is also specific to k and cannot be dropped. The resulting **quadratic discriminant function** is: $\delta_k(x) = -\frac{1}{2}(x-\mu_k)^\top\Sigma_k^{-1}(x-\mu_k) - \frac{1}{2}\log(|\Sigma_k|) + \log\pi_k$
This function is **quadratic** in x, which results in a curved decision boundary.
Kernel LDA: The Problem: When data is not linearly separable in the original input space, LDA will fail. The Kernel Trick: Data is mapped to a higher-dimensional *feature space* via a nonlinear transformation $\phi(x)$, where the classes become linearly separable. The LDA algorithm (Fisher's interpretation) depends on inner products of the data The **kernel trick** allows us to compute these inner products in the high-dimensional feature space without ever explicitly defining the mapping ϕ. We replace the inner product $\langle\phi(x_i), \phi(x_j)\rangle$ with a **kernel function** $k(x_i, x_j)$. **Example (Gaussian RBF Kernel):** $k(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{\sigma^2}\right)$
This allows Kernel LDA to find a linear boundary in the feature space, which corresponds to a highly non-linear boundary in the original input space.
The Validation Set Approach: Models: When comparing models, such as polynomials for predicting mpg from horsepower, we define a set of functions:
Linear (Degree 1): $\hat{f}_1(x) = \beta_0 + \beta_1(x)$
Quadratic (Degree 2): $\hat{f}_2(x) = \beta_0 + \beta_1(x) + \beta_2(x)^2$
Degree d: $\hat{f}_d(x) = \sum_{j=0}^{d}\beta_j(x)^j)$
Performance Metric: The model is fit on the training set. Its performance is evaluated on the validation set using a metric like Mean Squared Error (MSE): $MSE_{val} = \frac{1}{n_{val}}\sum_{i\in val}(y_i - \hat{f}(x_i))^2$ where n_{val} is the number of observations in the validation set, y_i is the true value, and $\hat{f}(x_i)$ is the prediction from the model fit on the training data.
Leave-One-Out Cross-Validation (LOOCV): LOOCV is a special case of K-fold CV where $K = n$, the number of observations.
Standard (Slow) Formula: The model must be fit n times. The LOOCV estimate of the test MSE is defined as: $CV_{(n)} = \frac{1}{n}\sum_{i=1}^{n}\text{MSE}_i = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i^{(i)})^2$ where $\hat{y}_i^{(i)}$ is the prediction for observation i from a model trained on all data *except* observation i. Shortcut (Fast) Formula for Linear Models: For linear regression models, $CV_{(n)}$ can be computed from a single model fit on the full dataset: $CV_{(n)} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{1 - h_i}\right)^2$ where $\hat{y}_i$ is the prediction from the model trained on all n data points, and h_i is the leverage statistic for observation i. **Leverage and the Hat Matrix**: In a linear model, $\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{e}$, the matrix of predictors is $\mathbf{X}$ and the response vector is $\mathbf{Y}$. **Fitted Values**: $\hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$
Hat Matrix: $(\mathbf{H})$: $\mathbf{H} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$. It maps $\mathbf{y}$ to $\hat{\mathbf{y}}$.
Leverage: The i-th diagonal element of $\mathbf{H}$.
$h_i = h_{ii} = \mathbf{x}_i^T(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{x}_i$
Leverage h_i measures the influence of observation i on its own predicted value. **Proof of the LOOCV Shortcut Formula**: The goal is to prove the identity: $e_{[i]} = y_i - \hat{y}_{[i]} = \frac{e_i}{1-h_i}$, where $e_i = y_i - \hat{y}_i$ is the standard residual.
K-Fold Cross-Validation:
The K-fold CV error is the average of the MSEs from K iterations: $CV_{(K)} = \frac{1}{K}\sum_{i=1}^{K}\text{MSE}_i$ where MSE_i is the mean squared error computed on the i-th fold when it served as the validation set. **Cross-Validation for Classification**: **Error Metric**: For classification, the test error is often the classification error rate. For LOOCV, the error for a single observation i is defined using an indicator function $I(\cdot)$: $Err_i = I(y_i \neq \hat{y}_i^{(i)})$ $Err_i = 1$ if the prediction is wrong, and 0 if correct. The n-fold (LOOCV) classification error is the average: $CV_{(n)} = \frac{1}{n}\sum_{i=1}^{n}Err_i$ **Logistic Regression Model**: A common model for classification is logistic regres-

sion, which models the probability of a class: $P(Y=1|X) = \frac{e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}{1+e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}} = \frac{1}{1+\exp(-\beta^T \mathbf{x})}$

Bias-Variance Trade-off in CV Choice:

LOOCV$(K=n)$:

Low Bias:
The model is trained on $n-1$ samples, which is very close to the full dataset n. The resulting error estimate $CV_{(n)}$ is nearly unbiased for the true test error. **High Variance**:
The n models are highly correlated (trained on almost identical data). Averaging n highly correlated estimates does not significantly reduce the overall variance. **K-Fold CV**$(K<n$, e.g., $K=5$ or $10)$:

High Bias:
The models are trained on smaller subsets (e.g., 90% of data for $K=10$). Since performance depends on n, this smaller training size means $CV_{(K)}$ tends to overestimate the true test error. **Low Variance**:
The K models are trained on less-overlapping data, making their error estimates less correlated. Averaging K less-correlated estimates results in a more stable final estimate.

The Bootstrap:
Bootstrap is a resampling technique used to estimate the uncertainty (e.g., standard error, confidence intervals) of a statistic. **Process**:
Given an original sample Z of size n.
Draw B "bootstrap samples" $(Z^{*1}, Z^{*2}, \ldots, Z^{*B})$, each of size n, by sampling *with replacement* from Z.
Compute the statistic of interest $\hat{\theta}$ for each bootstrap sample: $\hat{\theta}^{*j} = f(Z^{*j})$.
The collection $\{\hat{\theta}^{*j}\}_{j=1}^{B}$ is the bootstrap distribution, which serves as an empirical estimate of the true sampling distribution of $\hat{\theta}$.
Bootstrap Standard Error Estimate:
The standard error of the original statistic $\hat{\theta}$ is estimated by the standard deviation of the B bootstrap statistics:
$SE_{boot}(\hat{\theta}) = \sqrt{\frac{1}{B-1}\sum_{j=1}^{B}(\hat{\theta}^{*j} - \bar{\theta}^{*})^2}$
where $\bar{\theta}^{*} = \frac{1}{B}\sum_{j=1}^{B}\hat{\theta}^{*j}$.

Example: Portfolio Variance Minimization:
To find the α that minimizes $var(\alpha X + (1-\alpha)Y)$:
True Population Parameter (α): A function of true (co)variances. $\alpha = \frac{\sigma_Y^2 - \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}}$
Sample Statistic ($\hat{\alpha}$): An estimate based on sample (co)variances.
$\hat{\alpha} = \frac{\hat{\sigma}_Y^2 - \hat{\sigma}_{XY}}{\hat{\sigma}_X^2 + \hat{\sigma}_Y^2 - 2\hat{\sigma}_{XY}}$
Bootstrap Statistic ($\hat{\alpha}^*$): The same formula applied to a bootstrap sample.
We can find $\hat{SE}_{boot}(\hat{\alpha})$ by computing the standard deviation of B values of $\hat{\alpha}^*$. **Theoretical Justification**: The bootstrap method relies on an analogy. The (unknown) process of drawing a sample Z from the true population F to get $\hat{\theta}$ is mimicked by the (known) process of drawing a bootstrap sample Z^* from the empirical distribution $\hat{F}_n$ (i.e., the original sample) to get $\hat{\theta}^*$.
True World (Unknown): We want the sampling distribution of $(\hat{\theta} - \theta)$, where $Z \sim F$.
Bootstrap World (Knowable): We compute the distribution of $(\hat{\theta}^* - \hat{\theta})$, where $Z^* \sim \hat{F}_n$.
The core approximation is that the distributions of the errors are similar:
$\text{dist}\left(\sqrt{n}(\hat{\theta}^* - \hat{\theta})\right) \approx \text{dist}\left(\sqrt{n}(\hat{\theta} - \theta)\right)$

Mathematical Analysis of CV Error Overestimation:
CV Overestimation Theory:
We can model the test error $R(n)$ for a model trained on n samples as: $R(n) = R^* + \frac{c}{n}$
where R^* is the irreducible error and c/n is the reducible error (model variance), which decreases as n increases.
The true test error of the final model (trained on all n samples) is $R(n)$.
The k-fold CV error (CV_k) is the average error of k models, each trained on a smaller sample of size $n' = (\frac{k-1}{k})n$.
Therefore, CV_k is an estimate of $R(n')$: $CV_k \approx R(n') = R\left(\frac{k-1}{k}n\right) = R^* + \frac{c}{\left(\frac{k-1}{k}\right)n} = R^* + \frac{ck}{(k-1)n}$
Comparing CV_k to $R(n)$: $CV_k \approx R^* + \left(\frac{k}{k-1}\right)\frac{c}{n} > R^* + (1)\frac{c}{n} = R(n)$
Since $\frac{k}{k-1} > 1$, this shows that CV_k is a slightly biased estimate that, on average, overestimates the true test error $R(n)$. **A Flawed Correction Formula**:
If one makes the strong (and often false) assumption that the irreducible error $R^* = 0$, the relationship simplifies: $R(n) \approx \frac{c}{n}$, $CV_k \approx \frac{ck}{(k-1)n}$
Taking the ratio of these two approximations: $\frac{R(n)}{CV_k} \approx \frac{c/n}{ck/((k-1)n)} = \frac{c}{n}\cdot\frac{(k-1)n}{ck} = \frac{k-1}{k}$
This leads to a "corrected" CV estimate, $C\tilde{V}_k$: $R(n) \approx \left(\frac{k-1}{k}\right)\cdot CV_k = C\tilde{V}_k$
This correction is generally not used because its derivation relies on the false premise that $R^* = 0$. In real-world problems with noise ($R^* > 0$), the raw CV_k is a more reliable and conservative estimate of the test error.

Linear Model Selection:
Best Subset Selection (BSS):
Let M_0 be the null model (intercept only).
For $k = 1, \ldots, p$:
Fit all $\binom{p}{k}$ models with exactly k predictors.
Let M_k be the model with the smallest RSS (or largest R^2) among these.
Select the single best model from $M_0, \ldots, M_p$ using a criterion that balances fit and complexity (Adj. R^2, C_p, AIC, BIC, or Cross-Validation).
Computational Cost: 2^p total models must be fit.
Stepwise Selection Algorithms:
Forward Stepwise Selection (FSS)
Let M_0 be the null model.
For $k = 0, \ldots, p-1$:
Consider all $p-k$ models that add one additional predictor to M_k.
Let M_{k+1} be the model with the smallest RSS.
Select the best model from $M_0, \ldots, M_p$ using AIC, BIC, C_p, etc.
Computational Cost: $1 + \sum_{k=0}^{p-1}(p-k) = 1 + \frac{p(p+1)}{2}$ total models.
Backward Stepwise Selection (BSS)
Let M_p be the full model with all p predictors.
For $k = p, \ldots, 1$:
Consider all k models that remove one predictor from M_k.
Let M_{k-1} be the model with the smallest RSS.
Select the best model from $M_0, \ldots, M_p$ using AIC, BIC, C_p, etc.
Note: Cannot be used if $p > n$.
Model Selection Criteria
Basic Fit Metrics (Not for Selection)
Residual Sum of Squares (RSS): $RSS = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$. Always decreases as p increases.
R-squared (R^2): $R^2 = 1 - \frac{RSS}{TSS} = 1 - \frac{RSS}{\sum(y_i - \bar{y})^2}$. Always increases as p increases.
Complexity-Adjusted Criteria (For Selection) We want to select a model that minimizes (or maximizes) one of these. Let d be the number of predictors. **Adjusted R^2:** (Maximize) Adjusted $R^2 = 1 - \frac{RSS/(n-d-1)}{TSS/(n-1)}$ **Mallow's C_p:** (Minimize) $C_p = \frac{1}{n}(RSS + 2d\hat{\sigma}^2)$ where $\hat{\sigma}^2$ is an estimate of the error variance, typically from the full model. For OLS, C_p is equivalent to AIC. **Akaike Information Criterion (AIC):** (Minimize) $AIC = -2\log\mathcal{L}(\hat{\theta}) + 2d$
For a linear model with Gaussian errors, $\log\mathcal{L}(\hat{\theta})$ is proportional to $-RSS/(2\hat{\sigma}^2)$, so AIC simplifies: $AIC \propto \frac{1}{n\hat{\sigma}^2}(RSS + 2d\hat{\sigma}^2) \propto C_p$ **Bayesian Information Criterion (BIC):** (Minimize) $BIC = -2\log\mathcal{L}(\hat{\theta}) + \log(n)d$ For a linear model: $BIC \propto \frac{1}{n\hat{\sigma}^2}(RSS + \log(n)d\hat{\sigma}^2)$ BIC has a stronger penalty than AIC when $n \geq 8$ (since $\log(n) > 2$), tending to choose simpler models.
Theoretical Justification for AIC (K-L Divergence)
Goal: Find a model $\hat{p}$ that is "closest" to the true, unknown data generating process p.
Metric: Kullback-Leibler (K-L) Divergence. $K(p,\hat{p}) = \int p(y)\log\left(\frac{p(y)}{\hat{p}(y)}\right)dy = \mathbb{E}_p[\log(p(Y))] - \mathbb{E}_p[\log(\hat{p}(Y))]$ Minimizing $K(p,\hat{p})$ is equivalent to maximizing $\mathbb{E}_p[\log(\hat{p}(Y))]$.
$\log\mathcal{L}(\hat{\theta}) = \sum\log(\hat{p}(y_i))$ is a biased estimator for this quantity.
Akaike's Insight: The bias is approximately equal to $-d$ (the number of parameters). $\mathbb{E}[\log\mathcal{L}(\hat{\theta})] \approx \mathbb{E}_p[\log(\hat{p}(Y))] - d$
An approximately unbiased estimator for the target $\mathbb{E}_p[\log(\hat{p}(Y))]$ is $\log\mathcal{L}(\hat{\theta}) + d$.
AIC multiplies this by -2 for historical reasons: $AIC = -2\log\mathcal{L}(\hat{\theta}) - 2d$. (Note: some definitions use $+2d$, which is equivalent for minimization).
Shrinkage (Regularization) Methods
The general form is to minimize: Loss + Penalty.
$\hat{\beta} \leftarrow \arg\min_\beta\left(\|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \lambda\cdot P(\beta)\right)$
Ridge Regression (L2 Penalty)
Objective Function:
$\hat{\beta}^R = \arg\min_\beta\left(\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda\sum_{j=1}^{p}\beta_j^2\right)$
This is equivalent to $\min_\beta\left(\|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \lambda\|\beta\|_2^2\right)$.
Tuning Parameter λ:
If $\lambda = 0$, $\hat{\beta}^R = \hat{\beta}^{OLS}$. If $\lambda \to \infty$, $\hat{\beta}^R \to 0$. **Matrix Solution:** $\hat{\beta}^R = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$
The term $\lambda\mathbf{I}$ makes the matrix invertible, solving the problem of multicollinearity or $p > n$. **Properties:**
Standardization: Predictors $\mathbf{X}$ must be standardized before fitting.
Bias-Variance: Introduces bias ($\mathbb{E}[\hat{\beta}^R] \neq \beta$) to reduce variance.
Shrinkage: Shrinks coefficients towards 0, but does not set them to exactly 0 (unless $\lambda = \infty$).
Grouping Effect: With correlated predictors $x_j \approx x_k$, Ridge shrinks their coefficients $\hat{\beta}_j$ and $\hat{\beta}_k$ together.
LASSO (L1 Penalty) Objective Function (Penalized):
$\hat{\beta}^L = \arg\min_\beta\left(\frac{1}{2}\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda\sum_{j=1}^{p}|\beta_j|\right)$
This is equivalent to $\min_\beta\left(\|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \lambda'\|\beta\|_1\right)$.
Objective Function (Constrained):
$\hat{\beta}^L = \arg\min_\beta\left(\|\mathbf{y} - \mathbf{X}\beta\|_2^2\right)$ subject to $\sum_{j=1}^{p}|\beta_j| \leq s$
For every $\lambda \geq 0$, there is a corresponding $s \geq 0$ such that both problems yield the same solution.
Properties: Sparsity: The L_1 penalty can force coefficients to be **exactly zero**, performing automatic variable selection.
Geometric View: The L_1 constraint region $\|\beta\|_1 \leq s$ is a diamond (in 2D) or hyper-rhombus. The RSS contours (ellipses) are likely to make contact at a corner (vertex), where one or more $\beta_j = 0$. The L_2 Ridge constraint $\|\beta\|_2^2 \leq s$ is a circle/hypersphere, which has no corners, so $\beta_j \neq 0$.
Collinearity: If $x_j \approx x_k$, LASSO is unstable and tends to arbitrarily select one and set the other to 0.
Solution (Orthogonal Case): If $\mathbf{X}^T\mathbf{X} = \mathbf{I}$ (orthogonal design):
OLS Solution: $\hat{\beta}_j^{OLS} = \langle\mathbf{x}_j, \mathbf{y}\rangle$.
Ridge Solution: $\hat{\beta}_j^R = \hat{\beta}_j^{OLS}/(1+\lambda)$ (Proportional shrinkage).
Lasso Solution: $\hat{\beta}_j^L = \text{sign}(\hat{\beta}_j^{OLS})\left(|\hat{\beta}_j^{OLS}| - \lambda\right)_+$ (Soft-Thresholding).
This means if $|\hat{\beta}_j^{OLS}| \leq \lambda$, $\hat{\beta}_j^L = 0$.
Sign Consistency & Irrepresentable Condition:
LASSO can recover the true support $S = \{j : \beta_j \neq 0\}$ and $\text{sign}(\hat{\beta}_S) = \text{sign}(\beta_S)$ if the "Irrepresentable Condition" holds. This condition essentially states that the irrelevant features ($\mathbf{X}_{S^C}$) cannot be too highly correlated with the relevant features ($\mathbf{X}_S$).
$|\mathbf{X}_{S^C}^T\mathbf{X}_S(\mathbf{X}_S^T\mathbf{X}_S)^{-1}\text{sign}(\beta_S)|_\infty < 1$
Elastic Net (L1 and L2 Penalties)
Objective Function:
$\hat{\beta}^{EN} = \arg\min_\beta\left(\|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \lambda\left(\alpha\|\beta\|_1 + \frac{1-\alpha}{2}\|\beta\|_2^2\right)\right)$
Tuning Parameters:
$\lambda \geq 0$: Controls the total amount of regularization.
$\alpha \in [0,1]$: Controls the mix between Ridge and Lasso.
$\alpha = 0 \implies$ Ridge.
$\alpha = 1 \implies$ Lasso.
Properties:
Combines the strengths of both methods. Performs variable selection (sparsity) due to the L_1 part. Stably handles collinearity (grouping effect) due to the L_2 part. Can select more than n variables when $p > n$.
High-Dimensional Data ($p \gg n$)
The Problem: When $p > n$, the OLS solution is not unique. (X^TX) is singular. This is a state of "extreme multicollinearity."
The Trap: Standard metrics are misleading.
Training $R^2 \to 1$. Training $RSS \to 0$.
These metrics provide no evidence of good fit.
The Solution: Regularization (Ridge, Lasso, Elastic Net) is required.
The Rule: Models **must** be evaluated using Test Error (from a hold-out set) or Cross-Validation Error.
Curse of Dimensionality: Even with regularization, if the number of noise features is too large relative to the signal features, the true signal can be overwhelmed, and the Test Error will be high regardless.
Matrix Completion
Problem: Given a sparse matrix of observations $\mathbf{Y}$ (with observed set $\mathcal{O}$), find the underlying low-rank matrix $\mathbf{M}$ (where $\mathbf{Y} = \mathbf{M} + \mathbf{E}$).
Goal: Predict unobserved entries (Collaborative Filtering).
Parameterization: If $\text{rank}(\mathbf{M}) = r$, then $\mathbf{M} = \mathbf{U}\mathbf{V}^T$, where $\mathbf{U}$ is $d_1 \times r$ and $\mathbf{V}$ is $d_2 \times r$.
Degrees of Freedom: $r(d_1 + d_2 - r)$. We need $|\mathcal{O}|$ to be at least this large.
Hard Problem (Non-convex):
$\underset{\mathbf{N}}{\text{minimize}}\ \text{rank}(\mathbf{N})$ subject to $\mathcal{P}_\mathcal{O}(\mathbf{N}) = \mathcal{P}_\mathcal{O}(\mathbf{Y})$
Or (noisy version):
$\underset{\mathbf{N}}{\text{minimize}}\ \|\mathcal{P}_\mathcal{O}(\mathbf{Y} - \mathbf{N})\|_F^2$ This is NP-hard because $\text{rank}(\mathbf{N}) \leq r$
$\text{rank}(\cdot)$ is a non-convex function.
Convex Relaxation:
Vector Analogy: $\underset{}{\text{minimize}}\ \|\beta\|_0$ (non-convex) $\to$ $\underset{}{\text{minimize}}\ \|\beta\|_1$ (convex).
Matrix Analogy: $\underset{}{\text{minimize}}\ \text{rank}(\mathbf{X})$ (non-convex) $\to$ $\underset{}{\text{minimize}}\ \|\mathbf{X}\|_*$ (convex).
Nuclear Norm:
$\|\mathbf{X}\|_* = \sum_{i=1}^{\min(d_1,d_2)}\sigma_i(\mathbf{X})$ (where σ_i are the singular values of $\mathbf{X}$).
The Solvable Problem (Candès & Recht):
$\underset{\mathbf{X}}{\text{minimize}}\ \|\mathbf{X}\|_*$ subject to $\mathcal{P}_\mathcal{O}(\mathbf{X}) = \mathcal{P}_\mathcal{O}(\mathbf{Y})$
Recovery Guarantee: If $\mathbf{M}$ is low-rank (r) and $\mathcal{O}$ is sampled uniformly at random with $|\mathcal{O}| \geq C\cdot r(d_1+d_2)\log(d_1+d_2)$, then with high probability, the solution $\mathbf{X}$ to the convex problem is unique and $\mathbf{X} = \mathbf{M}$.
Overview of Non-linear Regression
Motivation
The primary limitation of the fundamental linear model is its assumption that the mean response must be a linear function of the inputs. In practice, this relationship often does not hold. For example, in the `Wage` vs. `Age` data, a simple linear regression line is a poor fit for the observed curved relationship.
General Model
To address this, various non-linear regression models are used. We begin with the general model:
$y_i = f(x_i) + \epsilon_i$ The goal is to find an estimate for the continuous function $f(\cdot)$. We "limit the search space" for $f(\cdot)$ by approximating it with a family of functions, such as polynomials, step functions, or splines.
Key Models
Key non-linear models include:
Polynomial regression, Step functions, Regression and Smoothing splines,
Local Regression, Generalized Additive Models (GAMs)
Polynomial Regression
Motivation and Formulation
Polynomial regression is an example of a "basis function approach" or "derived inputs approach". Instead of using the input x directly, it uses a set of derived polynomial terms as inputs in a linear regression.
The formulation for a polynomial regression model of degree p is: $y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \cdots + \beta_p x_i^p + \epsilon_i$
Key Properties and Challenges A crucial insight is that this model is just a **multiple linear regression model** with p inputs: $(x_i, x_i^2, \ldots, x_i^p)$. Because of this, all standard results and fitting methods from linear regression (e.g., least squares) still apply.
The main challenges are: **Choosing the degree (p):** A key problem is determining the appropriate degree. This is often done using **ANOVA (Analysis of Variance)** to compare nested models (e.g., degree 2 vs. 3 vs. 4) and checking p-values to see if higher-degree terms significantly improve the fit.
Local fit: This method can have difficulty fitting functions that are highly varying in local regions.
Boundary Behavior: High-degree polynomials can exhibit "wild behavior, especially near the tails" (boundaries) of the data.
Polynomial Logistic Regression
This approach can be combined with Generalized Linear Models (GLMs). For a binary response y_i, a logistic model with polynomial regression would be: $p(y_i = 1|x_i) = \frac{\exp(\beta_0+\beta_1 x_i+\cdots+\beta_p x_i^p)}{1+\exp(\beta_0+\beta_1 x_i+\cdots+\beta_p x_i^p)}$
This is solved as a typical logistic model, where one constructs a design matrix $\mathbf{X}$ with columns $(1, x_i, x_i^2, \ldots, x_i^p)$.
Step Functions (Piecewise Constants)
Formulation This method fits "piece-wise constants" by breaking the predictor's range into intervals.
A set of **cutpoints** $c_1, \ldots, c_p$ are created, dividing the data into $p+1$ intervals. A series of p "dummy" basis functions $c_k(x)$ are created: $c_k(x) = I(c_k \leq x < c_{k+1})$ where $I(\cdot)$ is an indicator function.
The model takes the form: $y_i = \beta_0 + \beta_1 c_1(x_i) + \cdots + \beta_p c_p(x_i) + \epsilon_i$ This is, once again, a **multiple linear regression model**.
Drawbacks The main challenges are determining the number and locations of the cutpoints. The resulting fit is also "non-smooth, not even continuous".
Regression Splines General Basis Function Concept Splines generalize both polynomial and step-function regression. We define a set of **basis functions**, $b_1(x), b_2(x), \ldots, b_p(x)$. The model is: $y_i = \beta_0 + \beta_1 b_1(x_i) + \cdots + \beta_p b_p(x_i) + \epsilon_i$. **Key Insight:** This is **always a multiple linear regression model**. Polynomials are a special case where $b_j(x) = x^j$. Step functions are a special case where $b_j(x) = I(c_j \leq x < c_{j+1})$.
Piecewise Polynomials and Constraints Splines are formally defined as **constrained piecewise polynomials**.
Method: Cut the predictor's range at "knots" (the name for cutpoints) and use a separate polynomial function on each sub-interval. **Problem:** An unconstrained piecewise polynomial will have "dis-continuity at knots." **Solution:** Add **constraints** to the least squares minimization to force smoothness. Force continuity (no jumps). Force continuous first derivative (no sharp corners). Force continuous second derivative (no abrupt changes in curvature). Each constraint added "reduces one degree of freedom" and model complexity. **Cubic Splines** A **spline function** of degree d is a piecewise polynomial of degree d that has continuous derivatives up to degree $d-1$ at its knots. The most common is a **Cubic Spline** (degree 3). It is a piecewise cubic polynomial that is continuous and has continuous 1st and 2nd derivatives at the knots. **Cubic Spline Degrees of Freedom (DOF)** A cubic spline with K knots has $\mathbf{K+4}$ degrees of freedom (DOF).
Initial Parameters: There are $K+1$ regions. Each is fit with a cubic polynomial, which has 4 parameters ($\beta_0, \beta_1, \beta_2, \beta_3$). Total initial parameters: $4 \times (K+1)$. **Constraints:** At each of the K knots, 3 constraints are applied (continuity, continuous 1st derivative, continuous 2nd derivative). Total constraints: $3 \times K$. **Total DOF** = (Total parameters) - (Total constraints) DOF $= 4(K+1) - 3K = 4K + 4 - 3K = K + 4$
Spline Basis Representation (Method 2) Instead of fitting with explicit constraints, we use a special set of **basis functions** in a multiple linear regression. The spline regression model is: $y_i = \beta_0 + \beta_1 b_1(x_i) + \cdots + \beta_{K+3} b_{K+3}(x_i) + \epsilon_i$
A common basis for a cubic spline is the **truncated power basis**. For K knots $(\xi_1, \ldots, \xi_K)$, this basis consists of $K+4$ features: $\{1, x, x^2, x^3, h(x,\xi_1), \ldots, h(x,\xi_K)\}$ Where $h(x,\xi)$ is the truncated power function: $h(x,\xi) = x - \xi^3 = \begin{cases}(x-\xi)^3 & \text{if } x > \xi \\ 0 & \text{otherwise}\end{cases}$ Using these features in a standard linear model *automatically* produces a fit that meets all the cubic spline constraints. **Natural Splines Motivation and Definition** A **Natural Spline** is a modified cubic spline designed to improve behavior at the boundaries (tails) of the data, where standard cubic splines can be "quite unstable".
Restriction: It adds a "further restriction near boundary" by requiring the function to be **linear** outside of the outer knots (i.e., on the intervals $(-\infty, \xi_1]$ and $[\xi_K, \infty)$). **Benefit:** Natural splines "generally behave better" and can produce narrower confidence intervals near the boundaries. **Natural Spline Degrees of Freedom (DOF)** The added linear constraints reduce the degrees of freedom from a standard cubic spline. **Cubic Spline DOF:** $K+4$. **Added Constraints:** The linearity constraint imposes 2 constraints at the first knot ξ_1 and 2 constraints at the last knot ξ_K, for a total of 4 additional constraints. **Natural Spline DOF** = (Cubic DOF) - (Boundary Constraints) DOF $= (K+4) - 4 = K$
A natural spline with K knots has K degrees of freedom. It is fit using a "variant of truncated power basis" with K total basis functions. **Model Selection and Comparison**
Choosing Knots Location: Knots can be "usually choose equally spaced" or placed more densely **Number (of Knots/DOF):** The best number of knots (or, equivalently, degrees of freedom) is typically determined using **cross-validation**.
Comparison: Splines vs. Polynomials Regression splines, particularly natural splines, "often give superior results to polynomial regression". **Flexibility:** Polynomials increase flexibility by increasing the degree p, which can "be dangerously in-approximate" and lead to "poor boundary behavior". Splines increase flexibility by increasing the number of knots K, while keeping the degree fixed (e.g., at 3). **Stability:** Natural splines are "much better" at handling boundaries. A degree-15 polynomial might exhibit "wild behavior... especially near the tails," while a natural cubic spline with the same 15 degrees of freedom remains stable.