Machine Learning: Module 1 Notes

Dr. Likhit Nayak

October 6, 2025

1 K-Nearest Neighbors (KNN) Algorithm

1.1 Core Concepts

1. **Non-parametric method:** KNN is a non-parametric algorithm, meaning it makes no assumptions about the underlying distribution of the data. The model structure is determined from the data itself, not by a pre-defined functional form.

2. **Neighborhood:** The core idea of KNN is that similar instances exist in close proximity. The "neighborhood" of a data point consists of the k closest data points in the training set.

3. **Euclidean Distance:** This is the most common metric for measuring closeness between two points. For $p, q \in \mathbb{R}^n$,
$$d(p,q) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}.$$
Other metrics (e.g., Manhattan, Minkowski) can also be used.

4. **Majority Vote (Classification):** In classification, a new data point is assigned the class label most common among its k nearest neighbors.

5. **Averaging (Regression):** In regression, the predicted value is the average of the target values of the k nearest neighbors.

6. **Choice of k:** The parameter k determines how many neighbors to consider. It significantly impacts performance and is typically chosen via cross-validation.

7. **Curse of Dimensionality:** As the number of features grows, the feature space becomes sparse, making distance metrics less meaningful and degrading local methods like KNN.

1.2 The KNN Algorithm

KNN is a memory-based (instance-based) learning algorithm and is considered a *lazy learner*: it stores the training data and performs computation at prediction time.

1.3 KNN for Classification

1. Choose a value for k.

2. Receive a new, unclassified data point.

3. Compute the distance (e.g., Euclidean) between the new point and all points in the training set.

4. Identify the k nearest neighbors (smallest distances).

5. Conduct a majority vote among their class labels.

6. Assign the class with the highest vote count to the new point.

1.4 KNN for Regression

1. Choose a value for k.
2. Receive a new data point for prediction.
3. Compute distances to all training points.
4. Identify the k nearest neighbors.
5. Predict by averaging their target values:
$$\hat{y} = \frac{1}{k}\sum_{i=1}^{k} y_{(i)}.$$
6. Assign this average as the prediction.

1.5 The Role of k and the Bias–Variance Tradeoff

The parameter k serves as a smoothing parameter:

- **Small k (e.g., $k = 1$):**
  - Model complexity: High (flexible, jagged decision boundary).
  - Bias: Low (captures fine patterns).
  - Variance: High (sensitive to noise, prone to overfitting).
- **Large k (e.g., $k = |$training set$|$):**
  - Model complexity: Low (smooth boundary).
  - Bias: High (may oversimplify, underfit).
  - Variance: Low (stable, not sensitive to individual points).

Optimal k balances bias and variance, commonly selected via cross-validation.

1.6 The Curse of Dimensionality

Local methods like KNN suffer when the number of dimensions p increases:

1. **Neighborhoods Become Less Local:** Points tend to be far from each other.
2. **Distance Metrics Lose Meaning:** Distances to nearest and farthest neighbors converge, reducing contrast.

2 Multiple Linear Regression

2.1 Key Concepts

- **Linear Model:** An approach for modeling the relationship between a response variable and one or more predictor variables using a linear equation.
- **Least Squares:** A standard method for fitting a model by minimizing the sum of the squared differences between the observed and predicted values.
- **Residual Sum of Squares (RSS):** The quantity that the least squares method aims to minimize.
- **Normal Equations:** The set of equations whose solution yields the least-squares coefficient estimates.
- **Matrix Notation:** A compact way to represent the linear model and its solution.
- **Fitted Values:** The predictions made by the model for the observed data points.

2.2 The Linear Model

The multiple linear regression model assumes a linear relationship between a set of p predictor variables $X = (X_1, X_2, \ldots, X_p)$ and a response variable Y:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p + \varepsilon.$$

This can be expressed as a function $f(X)$ for prediction:

$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j.$$

- β_0: The intercept, representing the expected value of Y when all X_j are zero.
- β_j $(j = 1, \ldots, p)$: The coefficients associated with each predictor, representing the average change in Y for a one-unit increase in X_j, holding other predictors constant.
- ε: The error term, assumed to have mean zero.

2.3 Method of Ordinary Least Squares (OLS)

We aim to find $\beta = (\beta_0, \beta_1, \ldots, \beta_p)$ that minimize the Residual Sum of Squares (RSS). A residual e_i is

$$e_i = y_i - \hat{y}_i = y_i - (\beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j).$$

The RSS is

$$\text{RSS} = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2.$$

The least squares estimates $\hat{\beta}$ minimize this sum.

2.3.1 Matrix Notation

Let y be the $n \times 1$ vector of responses, X the $n \times (p+1)$ design matrix (with a column of ones), β the $(p+1) \times 1$ coefficient vector, and ε the $n \times 1$ error vector. The model is

$$\hat{y} = X\beta,$$

and the RSS can be written as

$$\text{RSS}(\beta) = (y - X\beta)^\top (y - X\beta).$$

2.3.2 Derivation

To find the value of β that minimizes the RSS, we need to differentiate the RSS with respect to β and set the result to zero. This requires the use of matrix calculus.

$$\text{RSS}(\beta) = y^\top y - 2\beta^\top X^\top y + \beta^\top X^\top X \beta$$

Differentiating the first term: $y^\top y$ The term $y^\top y$ is a scalar that does not depend on β. Therefore, its derivative with respect to β is zero.

$$\frac{\partial}{\partial \beta}(y^\top y) = 0$$

Differentiating the second term: $-2\beta^\top X^\top y$ This term is a linear function of β. The rule for differentiating a scalar of the form $a^\top x$ with respect to the vector x is:

$$\frac{\partial(a^\top x)}{\partial x} = a$$

In our case, $x = \beta$ and $a = X^\top y$. Thus,

$$\frac{\partial}{\partial\beta}(-2\beta^\top X^\top y) = -2X^\top y$$

Differentiating the third term: $\beta^\top X^\top X\beta$ This term is a quadratic form in β. The rule for differentiating a quadratic form $x^\top Ax$ with respect to the vector x is:

$$\frac{\partial(x^\top Ax)}{\partial x} = (A + A^\top)x$$

In our case, $x = \beta$ and the matrix $A = X^\top X$. The matrix $X^\top X$ is symmetric, meaning $(X^\top X)^\top = X^\top X$. Therefore, the derivative simplifies to:

$$\frac{\partial}{\partial\beta}(\beta^\top X^\top X\beta) = 2(X^\top X)\beta$$

Combining the derivatives of all three terms, we get the derivative of the RSS with respect to β:

$$\frac{\partial \text{RSS}}{\partial\beta} = 0 - 2X^\top y + 2X^\top X\beta = -2X^\top y + 2X^\top X\beta$$

To find the minimum of the RSS, we set the derivative equal to zero:

$$-2X^\top y + 2X^\top X\beta = 0$$

We can simplify this equation by dividing by 2:

$$-X^\top y + X^\top X\beta = 0$$

Rearranging the terms of the above equation gives us:

$$X^\top X\beta = X^\top y$$

This is a system of linear equations that can be solved for the coefficient vector β. If the matrix $X^\top X$ is invertible (which is generally the case if the columns of X are linearly independent), we can pre-multiply both sides of the normal equations by the inverse of $X^\top X$:

$$(X^\top X)^{-1}(X^\top X)\hat{\beta} = (X^\top X)^{-1}X^\top y$$

Since $(X^\top X)^{-1}(X^\top X) = I$ (the identity matrix), we are left with the solution for $\hat{\beta}$:

$$\hat{\beta} = (X^\top X)^{-1}X^\top y$$

This final expression is the Ordinary Least Squares (OLS) estimator for the coefficient vector β.

3 Ridge Regression

Ridge regression is a regularization technique used to address common problems that arise in multiple linear regression, particularly when dealing with multicollinearity or when the number of predictors (p) is large relative to the number of observations (n). It improves upon the Ordinary Least Squares (OLS) estimator by introducing a small amount of bias to achieve a significant reduction in variance, often leading to a lower overall prediction error. Here are the drawbacks of least squares method:

1. **High Variance:** Multicollinearity makes $(X^\top X)^{-1}$ unstable, yielding high-variance $\hat{\beta}$.
2. **Impossibility when $p > n$:** $X^\top X$ singular, no unique solution.

3.1 Key Concepts

- **Shrinkage:** Shrinking coefficient estimates towards zero to reduce variance.
- **L2 Penalty:** Sum of squared coefficients, used in Ridge Regression.
- **Regularization Parameter (λ):** Controls strength of the penalty.
- **Coefficient Path:** How estimates change as λ varies.

3.2 Why Shrinkage?

Shrinkage is the central mechanism of Ridge Regression. The term refers to the process of pulling, or "shrinking," the estimated regression coefficients towards zero. In least squares, coefficient estimates are solely determined by minimizing the RSS, which can lead to very large and unstable estimates if predictors are correlated. These large estimates have high variance, meaning they would change dramatically if a different training sample were used. By introducing the penalty term, Ridge regression forces the model to find a balance between two competing goals:

1. **Goodness-of-fit:** Minimizing the RSS to fit the training data well.
2. **Model Simplicity/Stability:** Keeping the magnitude of the coefficients small.

This process results in coefficient estimates that are systematically smaller (in absolute value). This reduction in magnitude makes the model less sensitive to the noise in the training data, thereby reducing the model's variance. While this introduces a small amount of bias (the estimates are no longer unbiased), the reduction in variance often outweighs the increase in bias, leading to a model that generalizes better to unseen data.

3.3 Ridge Regression

The specific form of the penalty used in Ridge Regression is the **L2 penalty**, given by $\lambda \sum_{j=1}^{p} \beta_j^2$. It is also known as the squared Euclidean norm or L2-norm of the coefficient vector, scaled by λ.

$$\mathrm{RSS}_\lambda = \sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2,$$

where $\lambda \geq 0$ is chosen by cross-validation. In matrix form:

$$\mathrm{RSS}_\lambda = (y - X\beta)^\top (y - X\beta) + \lambda \beta^\top \beta.$$

Setting derivative to zero yields

$$\hat{\beta}_{\text{ridge}} = (X^\top X + \lambda I)^{-1} X^\top y.$$

The **regularization parameter** $\lambda \geq 0$ is a hyperparameter that controls the strength of the penalty term. It dictates the trade-off between the fit to the data (RSS) and the magnitude of the coefficients (the penalty). As $\lambda \to 0$, $\hat{\beta}_{\text{ridge}} \to \hat{\beta}_{\text{ols}}$, while as $\lambda \to \infty$, $\hat{\beta}_{\text{ridge}} \to 0$.

4 Lasso Regression

Lasso, which stands for *Least Absolute Shrinkage and Selection Operator*, is a regularization technique used in statistical modeling and machine learning. Like Ridge Regression, it aims to improve the prediction accuracy and interpretability of regression models by shrinking the regression coefficients. However, Lasso has a distinct property that sets it apart: it can perform automatic variable selection by forcing some of the coefficient estimates to be exactly zero.

4.1 Key Concepts

1. **L1 Penalty:** The core of Lasso is its penalty term, which is the sum of the absolute values of the coefficients (the L1-norm). This penalty is directly responsible for its variable selection property.
2. **Sparsity:** A model is considered 'sparse' if many of its coefficients are exactly zero. Lasso produces sparse models by eliminating less important features from the model entirely. This makes the final model simpler and easier to interpret.
3. **Soft-Thresholding:** This is the mathematical operation that results from the L1 penalty. Unlike Ridge regression, which shrinks all coefficients proportionally towards zero, Lasso's soft-thresholding shrinks coefficients by a constant amount and sets any coefficient to zero if it falls below a certain threshold. This is the mechanism that generates the zero coefficients.

4.2 The Lasso Formulation

The objective is to minimize the Residual Sum of Squares (RSS) subject to a constraint, where the constraint is incorporated into the objective function as L1 penalty term:

$$\mathrm{RSS}_\lambda = \sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_j |\beta_j|,$$

In this form:

- The term $\lambda \sum_j |\beta_j|$ is the *L1 penalty*.
- $\lambda \geq 0$ is the tuning parameter that controls the strength of the penalty. As λ increases, the penalty becomes more severe, leading to greater shrinkage and more coefficients being forced to zero. When $\lambda = 0$, the penalty has no effect, and the Lasso solution is identical to the Ordinary Least Squares (OLS) solution.

4.3 Key Difference: Lasso (L1) vs. Ridge (L2) Penalty

- **Lasso Penalty (L1-norm):** $\lambda \sum_j |\beta_j|$ (sum of absolute values).
- **Ridge Penalty (L2-norm):** $\lambda \sum_j \beta_j^2$ (sum of squared values).

This seemingly small difference has a profound impact:

- **Variable Selection:** The L1 penalty can shrink coefficients all the way to *exactly zero*, effectively removing variables from the model. The L2 penalty shrinks coefficients *asymptotically towards zero* but never sets them exactly to zero (unless they were already zero).
- **Model Sparsity:** Consequently, Lasso produces sparse and more interpretable models, while Ridge retains all predictors in the final model.

4.3.1 Properties of the Lasso and Comparison with Ridge

Lasso (Least Absolute Shrinkage and Selection Operator) and Ridge Regression are both regularization techniques used to prevent overfitting in linear models. They achieve this by adding a penalty term to the ordinary least squares (OLS) objective function. The key difference lies in the type of penalty used, which leads to distinct properties.

$$\begin{aligned}
\text{OLS Objective:} \quad & \min_{\beta_0,\beta} \sum_i (y_i - \beta_0 - \sum_j x_{ij}\beta_j)^2, \\
\text{Ridge Objective:} \quad & \min_{\beta_0,\beta} \sum_i (y_i - \beta_0 - \sum_j x_{ij}\beta_j)^2 + \lambda \sum_j \beta_j^2, \\
\text{Lasso Objective:} \quad & \min_{\beta_0,\beta} \sum_i (y_i - \beta_0 - \sum_j x_{ij}\beta_j)^2 + \lambda \sum_j |\beta_j|.
\end{aligned}$$

4.4 Power of Lasso in High Dimensions

It is often assumed that even though we have a vast number of potential predictors, only a small fraction of them are truly related to the outcome. For example, in genomics, a disease might be influenced by a handful of genes out of tens of thousands measured. This is known as the sparsity assumption. Without this assumption, it would be statistically impossible to disentangle the signals of so many predictors from a limited number of samples.

Lasso is exceptionally powerful in high-dimensional, sparse settings precisely because it is designed to handle this structure. It performs two critical tasks simultaneously:

1. **Regularization:** It shrinks coefficients to prevent overfitting, which is a major risk when $p > n$.
2. **Variable Selection:** It automatically identifies a smaller, more manageable subset of predictors by forcing the coefficients of irrelevant ones to zero.

5 Logistic Regression

The primary goal of the logistic regression model is to predict the probability that a given input $\mathbf{x}$ belongs to a specific class k out of K possible classes. These are called the posterior probabilities, denoted as $Pr(G = k|X = x)$. For these probabilities to be valid, they must satisfy two key conditions:

- Each probability must be between 0 and 1 (i.e., remain in $[0, 1]$).
- The sum of the probabilities for all K classes for a given input $\mathbf{x}$ must equal 1.

The model aims to achieve this by using a linear function of the input $\mathbf{x}$ in the form of $\beta_0 + \beta^T x$. However, a linear function's output can be any real number, not just a value between 0 and 1. This is where the central idea of logistic regression comes in.

5.1 Log-Odds (Logit) Transformation

Instead of modeling the probabilities directly with a linear function, the logistic regression model models the log-odds (also known as the logit) as a linear function of $\mathbf{x}$.

- **Odds:** The odds of an event is the ratio of the probability of the event occurring to the probability of it not occurring. For a class k, the odds would be:

$$\frac{Pr(G = k|X = x)}{1 - Pr(G = k|X = x)}.$$

- **Log-Odds (Logit):** This is simply the natural logarithm of the odds. The logit transformation is crucial because it takes a value that is in $[0, 1]$ (a probability) and maps it to the entire real number line $(-\infty, +\infty)$. This allows for a linear model to be used.

5.2 Extending to Multiple Classes ($K > 2$)

When there are more than two classes, the model needs to handle K probabilities. To do this while ensuring the probabilities sum to one, the model selects one class to be a reference category (in this text, class K is chosen as the denominator). It then models the log-odds of each of the other $K - 1$ classes relative to this reference class.

This is what is shown in the first set of equations:

$$\log\left(\frac{Pr(G=1|X=x)}{Pr(G=K|X=x)}\right) = \beta_{10} + \beta_1^T x$$
$$\log\left(\frac{Pr(G=2|X=x)}{Pr(G=K|X=x)}\right) = \beta_{20} + \beta_2^T x$$
$$\vdots$$
$$\log\left(\frac{Pr(G=K-1|X=x)}{Pr(G=K|X=x)}\right) = \beta_{(K-1)0} + \beta_{K-1}^T x$$

Each equation represents a linear model for the log-odds of being in that specific class versus the reference class K. There are $K-1$ such equations because the probability of the K-th class is implicitly defined by the fact that all probabilities must sum to one.

The choice of the reference class is arbitrary and the estimates are equivariant under this choice. This means that while the specific coefficient values (β) will change if you pick a different reference class, the final predicted probabilities for each class will remain the same.

5.3 Derivation of the Formulas for the Posterior Probabilities

Let's start from the $K-1$ logit equations, using class K as the reference.

5.3.1 Step 1: Exponentiate the logit equations

The initial equations are of the form:

$$\log\left(\frac{Pr(G=k|X=x)}{Pr(G=K|X=x)}\right) = \beta_{k0} + \beta_k^T x$$

To remove the logarithm, we take the exponential of both sides for each of the $K-1$ equations:

$$\frac{Pr(G=k|X=x)}{Pr(G=K|X=x)} = \exp(\beta_{k0} + \beta_k^T x)$$

5.3.2 Step 2: Isolate the probabilities of the non-reference classes

For each k from 1 to $K-1$, we can express its probability in terms of the reference class's probability:

$$Pr(G=k|X=x) = Pr(G=K|X=x) \cdot \exp(\beta_{k0} + \beta_k^T x)$$

5.3.3 Step 3: Use the constraint that all probabilities sum to one

The sum of probabilities over all K classes must be 1:

$$\sum_{j=1}^{K} Pr(G=j|X=x) = 1$$

Let's expand this sum:

$$Pr(G=1|X=x) + Pr(G=2|X=x) + \cdots + Pr(G=K-1|X=x) + Pr(G=K|X=x) = 1$$

Now, substitute the expressions from Step 2 into this equation:

$$(Pr(G=K|X=x) \cdot \exp(\beta_{10} + \beta_1^T x)) + \cdots + (Pr(G=K|X=x) \cdot \exp(\beta_{(K-1)0} + \beta_{K-1}^T x)) \\ + Pr(G=K|X=x) = 1$$

5.3.4 Step 4: Solve for the reference class probability, $Pr(G = K|X = x)$

Factor out $Pr(G = K|X = x)$ from the left side:

$$Pr(G = K|X = x) \cdot (\exp(\beta_{10} + \beta_1^T x) + \cdots + \exp(\beta_{(K-1)0} + \beta_{K-1}^T x) + 1) = 1$$

Using summation notation, this is:

$$Pr(G = K|X = x) \cdot \left(1 + \sum_{l=1}^{K-1} \exp(\beta_{l0} + \beta_l^T x)\right) = 1$$

Now, isolate $Pr(G = K|X = x)$ to get its final formula:

$$Pr(G = K|X = x) = \frac{1}{1 + \sum_{l=1}^{K-1} \exp(\beta_{l0} + \beta_l^T x)}$$

5.3.5 Step 5: Solve for the non-reference class probabilities

Go back to the equation from Step 2:

$$Pr(G = k|X = x) = Pr(G = K|X = x) \cdot \exp(\beta_{k0} + \beta_k^T x)$$

Now substitute the formula we just derived for $Pr(G = K|X = x)$ into this equation:

$$Pr(G = k|X = x) = \left[\frac{1}{1 + \sum_{l=1}^{K-1} \exp(\beta_{l0} + \beta_l^T x)}\right] \cdot \exp(\beta_{k0} + \beta_k^T x)$$

This gives us the final formula for the non-reference classes ($k = 1, \ldots, K - 1$):

$$Pr(G = k|X = x) = \frac{\exp(\beta_{k0} + \beta_k^T x)}{1 + \sum_{l=1}^{K-1} \exp(\beta_{l0} + \beta_l^T x)}$$

5.4 How will the above change for $K = 2$?

When $K = 2$, the situation simplifies dramatically. Your classes are just $G = 1$ and $G = 2$. Let's choose $G = 2$ as the reference class. Since there is only one non-reference class ($K - 1 = 1$), you only need one equation:

$$\log\left(\frac{Pr(G = 1|X = x)}{Pr(G = 2|X = x)}\right) = \beta_{10} + \beta_1^T x$$

Let's apply the general formulas we derived, setting $K = 2$. The summation $\sum_{l=1}^{K-1}$ just becomes a single term where $l = 1$. For the reference class ($k = 2$):

$$Pr(G = 2|X = x) = \frac{1}{1 + \exp(\beta_{10} + \beta_1^T x)}$$

For the non-reference class ($k = 1$):

$$Pr(G = 1|X = x) = \frac{\exp(\beta_{10} + \beta_1^T x)}{1 + \exp(\beta_{10} + \beta_1^T x)}$$

These are precisely the formulas for the standard binary logistic regression model, where the probability of one class is given by the sigmoid (or logistic) function of the linear model, and the probability of the other class is just one minus that value. You can easily verify that $Pr(G = 1|X = x) + Pr(G = 2|X = x) = 1$.

5.5 Fitting the Model: Maximum Likelihood Estimation

We estimate the coefficients of logistic regression, β, by maximizing the likelihood of the observed data. This approach is called Maximum Likelihood Estimation (MLE). For a binary classification problem, we model the outcome of each observation as a Bernoulli trial. Let $y_i \in \{0, 1\}$ be the observed class for observation i, and let $p(x_i) = P(Y = 1|X = x_i; \beta)$ be the probability of the positive class predicted by our model. The probability of observing the specific outcome y_i can be written compactly using the probability mass function of the Bernoulli distribution:

$$P(y_i|x_i; \beta) = p(x_i)^{y_i}(1 - p(x_i))^{1-y_i}$$

This expression works for both cases: if the true label $y_i = 1$, the expression simplifies to $p(x_i)$, and if $y_i = 0$, it simplifies to $1 - p(x_i)$.

Likelihood function Assuming all N observations in the dataset are independent, the total likelihood function $L(\beta)$ is the product of the individual probabilities:

$$L(\beta) = \prod_{i=1}^{N} P(y_i|x_i; \beta) = \prod_{i=1}^{N} p(x_i)^{y_i}(1 - p(x_i))^{1-y_i}$$

The goal of MLE is to find the parameter values β that maximize this likelihood function.

Log-likelihood function For mathematical convenience, it is easier to maximize the natural logarithm of the likelihood function, called the log-likelihood $\ell(\beta)$. Since the logarithm is a monotonically increasing function, maximizing the log-likelihood is equivalent to maximizing the likelihood. The log-likelihood is:

$$\ell(\beta) = \log L(\beta) = \sum_{i=1}^{N} [y_i \log p(x_i) + (1 - y_i) \log(1 - p(x_i))]$$

In machine learning, we typically frame optimization problems as minimizing a cost or loss function. Maximizing the log-likelihood is equivalent to minimizing the negative log-likelihood. By convention, we use the average negative log-likelihood as the final objective function $J(\beta)$:

$$J(\beta) = -\frac{1}{N}\ell(\beta) = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log p(x_i) + (1 - y_i) \log(1 - p(x_i))]$$

This objective function is known as the **Binary Cross-Entropy Loss**. There is no closed-form solution for the β that minimizes this function, so it is optimized using numerical methods like gradient descent.

6 Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis is another method for classification that, unlike logistic regression which models the posterior probabilities directly, models the distribution of the predictors X separately in each of the response classes. It then uses Bayes' theorem to derive the posterior probabilities.

6.1 Key Concepts

- **Bayes' Rule for Classification:**

  $$P(Y = k \mid X = x) = \frac{p(x \mid Y = k)\, P(Y = k)}{p(x)},$$

  and classification chooses $\arg\max_k p(x \mid Y = k)\, \pi_k$.

- **Class-Conditional Densities:** Each class k assumed Gaussian with mean μ_k and common covariance Σ.

- **Decision Boundary:** Hyperplane where $Pr(G = k|X = \boldsymbol{x}) = Pr(G = l|X = \boldsymbol{x})$.

6.2 Assumptions

The core assumption of LDA is that we model each class density, $f_k(\boldsymbol{x})$, as a multivariate Gaussian distribution with a class-specific mean vector $\boldsymbol{\mu_k}$ and a covariance matrix that is **common** to all classes, $\boldsymbol{\Sigma}$.

$$f_k(\boldsymbol{x}) = \frac{1}{(2\pi)^{p/2}|\boldsymbol{\Sigma}|^{1/2}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu_k})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu_k})}$$

6.3 Derivation

Let $f_k(\boldsymbol{x})$ be the class-conditional density of X in class $G = k$, and let π_k be the prior probability of class k, with $\sum_{k=1}^{K} \pi_k = 1$. To understand the decision boundary between any two classes, k and l, we can examine the log-ratio of their posterior probabilities. A point $\boldsymbol{x}$ is assigned to class k over class l if $Pr(G = k|X = \boldsymbol{x}) > Pr(G = l|X = \boldsymbol{x})$, which is equivalent to their log-ratio being positive.

$$\begin{aligned}\log\frac{Pr(G = k|X = \boldsymbol{x})}{Pr(G = l|X = \boldsymbol{x})} &= \log\frac{f_k(\boldsymbol{x})\pi_k}{f_l(\boldsymbol{x})\pi_l} \\ &= \log\frac{f_k(\boldsymbol{x})}{f_l(\boldsymbol{x})} + \log\frac{\pi_k}{\pi_l}\end{aligned}$$

Now, we substitute the multivariate Gaussian density for $f_k(\boldsymbol{x})$ and $f_l(\boldsymbol{x})$. The constant term $(2\pi)^{p/2}|\boldsymbol{\Sigma}|^{1/2}$ cancels out because the covariance matrix $\boldsymbol{\Sigma}$ is assumed to be the same for all classes. We are left with the difference of the quadratic terms:

$$\begin{aligned}\log\frac{f_k(\boldsymbol{x})}{f_l(\boldsymbol{x})} &= -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu_k})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu_k}) + \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu_l})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu_l}) \\ &= -\frac{1}{2}(\boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{x} - 2\boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu_k} + \boldsymbol{\mu_k}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu_k}) \\ &\quad + \frac{1}{2}(\boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{x} - 2\boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu_l} + \boldsymbol{\mu_l}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu_l})\end{aligned}$$

The quadratic terms $\boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{x}$ cancel each other out. This is a crucial step that results from the equal covariance assumption. Rearranging the remaining terms, we get:

$$\begin{aligned}\log\frac{f_k(\boldsymbol{x})}{f_l(\boldsymbol{x})} &= \boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu_k} - \boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu_l} - \frac{1}{2}\boldsymbol{\mu_k}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu_k} + \frac{1}{2}\boldsymbol{\mu_l}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu_l} \\ &= \boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu_k}-\boldsymbol{\mu_l}) - \frac{1}{2}(\boldsymbol{\mu_k}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu_k} - \boldsymbol{\mu_l}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu_l})\end{aligned}$$

Plugging this back into the log-ratio of posteriors gives the final expression:

$$\log\frac{Pr(G = k|X = \boldsymbol{x})}{Pr(G = l|X = \boldsymbol{x})} = \log\frac{\pi_k}{\pi_l} - \frac{1}{2}(\boldsymbol{\mu_k}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu_k} - \boldsymbol{\mu_l}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu_l}) + \boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu_k}-\boldsymbol{\mu_l})$$

This equation is a linear function of $\boldsymbol{x}$. This linear log-odds function implies that the decision boundary between classes k and l—the set of points where $Pr(G = k|X = \boldsymbol{x}) = Pr(G = l|X = \boldsymbol{x})$ (i.e., the log-ratio is zero)—is linear in $\boldsymbol{x}$. In a p-dimensional space, this boundary is a hyperplane.

7 Model & Feature Selection

7.1 Classical Subset Selection Methods

Best-Subset Selection

1. Fit all 2^p possible submodels.
2. For each size k, choose the model with lowest RSS.
3. Use BIC, or cross-validation to pick among sizes.

Forward-Stepwise Selection

1. Start with the null model.
2. At each step, add the predictor that most decreases RSS.
3. Continue until all predictors are included; select best by a criterion.

Backward-Stepwise Selection

1. Start with the full model.
2. At each step, remove the predictor whose removal least increases RSS.
3. Continue until no predictors remain; select best.

7.2 Pros and Cons

- Best-Subset: Optimal but $O(2^p)$.
- Stepwise: Greedy, $O(p^2)$, not guaranteed optimal.