Lecture 2: Probabilistic parameter estimation and Bayes Decision Rule

Course: 2110573 Pattern Recognition

2026-01-20
version 1.0

1 Recap of Linear Regression

Linear regression is a supervised learning algorithm used to predict a continuous output. The model, or hypothesis, is a linear function of the input features.

1.1 Hypothesis Representation

For a single training example $\mathbf{x}_i$, the hypothesis $h_\theta(\mathbf{x}_i)$ is given by:

$$h_\theta(\mathbf{x}_i) = \theta_0 + \theta_1 x_{i,1} + \theta_2 x_{i,2} + \cdots + \theta_n x_{i,n} \tag{1}$$

where the θ_j values are the parameters (weights) of the model. By assuming a feature $x_{i,0} = 1$, we can write this in a more compact vector form:

$$h_\theta(\mathbf{x}_i) = \sum_{j=0}^{n} \theta_j x_{i,j} = \boldsymbol{\theta}^T \mathbf{x}_i \tag{2}$$

Here, $\boldsymbol{\theta}$ and $\mathbf{x}_i$ are column vectors.

1.2 Cost Function

To find the best parameters $\boldsymbol{\theta}$, we need to minimize a cost function. A common choice for linear regression is the Mean Squared Error (MSE), also known as the L2 loss. The cost function $J(\boldsymbol{\theta})$ is:

$$J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m}(y_i - h_\theta(\mathbf{x}_i))^2 = \frac{1}{m}\sum_{i=1}^{m}(y_i - \boldsymbol{\theta}^T\mathbf{x}_i)^2 \tag{3}$$

where m is the number of training examples.

1.3 Solving for Parameters

There are two primary methods to find the $\boldsymbol{\theta}$ that minimizes $J(\boldsymbol{\theta})$:

1. Gradient Descent This is an iterative method where we update the parameters in the opposite direction of the gradient of the cost function. The update rule for each parameter θ_j is:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) \tag{4}$$

For MSE, this becomes:

$$\theta_j := \theta_j + r \sum_{i=1}^{m}(y_i - \boldsymbol{\theta}^T\mathbf{x}_i)x_i^{(j)} \tag{5}$$

(Here, r is the learning rate, equivalent to α divided by m).

2. Normal Equation (Closed-Form Solution) This method solves for $\boldsymbol{\theta}$ directly without iteration.

$$\boldsymbol{\theta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{6}$$

where $\mathbf{X}$ is the design matrix (with each row being a training example $\mathbf{x}_i^T$) and $\mathbf{y}$ is the vector of all target values.

1.4 Learning Rate Selection

The learning rate, denoted by r (or α), is a critical hyperparameter in iterative optimization algorithms like gradient descent. The parameter update rule is given by:

$$\theta_j \leftarrow \theta_j + r\sum_{i=1}^{m}(y_i - h_\theta(x_i))x_i^{(j)}$$

The choice of learning rate has a significant impact on model training:

- **Too small**: The model converges very slowly, as it takes tiny steps towards the minimum of the loss function.
- **Too large**: The model may fail to converge, as the steps overshoot the minimum, causing the loss to oscillate or even diverge.

Best Practices:

- Typically, the learning rate is normalized by the size of the mini-batch (m).
- Common values for r range from 0.1 to 0.001.
- It is often beneficial to use a learning rate schedule, where the learning rate decays over time, allowing for larger steps at the beginning of training and finer adjustments as the model approaches convergence.

1.5 Scaling Input Data

The scale of input features can significantly affect the training process. Features with vastly different ranges or variances (e.g., age [0-80] vs. gender [0,1]) can make parameter initialization and learning rate selection difficult. The model's hypothesis, $h_\theta(x) = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n$, can be sensitive to this variance imbalance.

Solution: Scale all input features to be in a similar range. Common scaling techniques include:

- **Normalization**: Scale data to the range [0, 1] or [-1,1].
- **Standardization**: Scale data to have a mean of 0 and a standard deviation of 1 (standard normal distribution).

Important: The scaling statistics (e.g., mean and standard deviation) must be computed from the training data only and then applied consistently to the validation and test data.

1.6 Feature Selection

Often, using a smaller subset of the most relevant features can lead to better model performance, reduced overfitting, and faster training. Identifying the best features is not trivial. Common approaches include:

- Cross-validation
- Random Forest feature importance
- Boosting methods

1.7 Early Stopping

As training progresses, the model's bias on the training data decreases, but its variance may start to increase, leading to overfitting. To prevent this, we monitor the model's performance on a separate **validation set**. Training should be stopped when the error on the validation set begins to increase, as this indicates that the model is starting to lose its ability to generalize.

2 Probabilistic Interpretation of Linear Regression

We can justify the use of MSE through a probabilistic approach using the principle of **Maximum Likelihood Estimation (MLE)**.

Let's assume that the target variable y_i is related to the features by our hypothesis plus some random error term ϵ_i:

$$y_i = \boldsymbol{\theta}^T \mathbf{x}_i + \epsilon_i \tag{7}$$

A key assumption is that this error ϵ_i is independently and identically distributed (i.i.d.) according to a Gaussian (Normal) distribution with a mean of 0 and some variance σ^2, i.e., $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$.

This implies that the probability of y_i given $\mathbf{x}_i$ and parameterized by $\boldsymbol{\theta}$ is also normally distributed:

$$p(y_i|\mathbf{x}_i; \boldsymbol{\theta}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \boldsymbol{\theta}^T \mathbf{x}_i)^2}{2\sigma^2}\right) \tag{8}$$

The likelihood of the entire dataset is the product of these probabilities for all training examples:

$$L(\boldsymbol{\theta}) = \prod_{i=1}^{m} p(y_i|\mathbf{x}_i; \boldsymbol{\theta}) \tag{9}$$

Maximizing the likelihood is equivalent to maximizing the log-likelihood $l(\boldsymbol{\theta}) = \log L(\boldsymbol{\theta})$:

$$l(\boldsymbol{\theta}) = \sum_{i=1}^{m} \log p(y_i|\mathbf{x}_i; \boldsymbol{\theta}) \tag{10}$$

After substituting the Gaussian PDF and simplifying, maximizing $l(\boldsymbol{\theta})$ is equivalent to minimizing:

$$\sum_{i=1}^{m} (y_i - \boldsymbol{\theta}^T \mathbf{x}_i)^2 \tag{11}$$

This is exactly the MSE cost function. Therefore, minimizing MSE is equivalent to finding the Maximum Likelihood Estimate for $\boldsymbol{\theta}$ under the assumption of normally distributed errors.

3 Logistic Regression

Logistic regression is used for binary classification problems, where the output y is either 0 or 1.

3.1 Hypothesis Representation

Since the output of linear regression can be any real number, it is not suitable for classification. We can constrain the output to be between 0 and 1 by passing the linear combination $\boldsymbol{\theta}^T \mathbf{x}$ through the **logistic function** (or sigmoid function), $g(z)$:

$$g(z) = \frac{1}{1 + e^{-z}} \tag{12}$$

The hypothesis for logistic regression is then:

$$h_\theta(\mathbf{x}) = g(\boldsymbol{\theta}^T \mathbf{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^T \mathbf{x}}} \tag{13}$$

This output can be interpreted as the probability that $y = 1$ given $\mathbf{x}$.

3.2 Conditional Probability

In the context of logistic regression, we assume that the target variable $y_i \in \{0, 1\}$ follows a Bernoulli distribution. The model's hypothesis, $h_\theta(\mathbf{x}_i)$, represents the probability that the output is 1. Specifically, we define:

$$\begin{aligned} P(y_i = 1|\mathbf{x}_i; \boldsymbol{\theta}) &= h_\theta(\mathbf{x}_i) \\ P(y_i = 0|\mathbf{x}_i; \boldsymbol{\theta}) &= 1 - h_\theta(\mathbf{x}_i) \end{aligned}$$

To simplify the mathematical treatment of the likelihood function, these two cases are combined into a single expression, known as the probability mass function for a Bernoulli trial:

$$P(y_i|\mathbf{x}_i; \boldsymbol{\theta}) = (h_\theta(\mathbf{x}_i))^{y_i}(1 - h_\theta(\mathbf{x}_i))^{1-y_i} \tag{14}$$

This compact form is particularly useful because when $y_i = 1$, the second term $(1 - h_\theta(\mathbf{x}_i))^0$ becomes 1, and when $y_i = 0$, the first term $(h_\theta(\mathbf{x}_i))^0$ becomes 1. This allows us to take the product of probabilities across all m training examples to form the Likelihood function $L(\boldsymbol{\theta})$.

3.3 Probabilistic Interpretation and Update Rule

We start with the log-likelihood function, $l(\boldsymbol{\theta})$, which we want to maximize. The process of finding the parameters that maximize this function is called Maximum Likelihood Estimation (MLE).

The log-likelihood is given by:

$$\begin{aligned} l(\boldsymbol{\theta}) &= \log L(\boldsymbol{\theta}) \\ &= \log \prod_{i=1}^{m} P(y_i|\mathbf{x}i; \boldsymbol{\theta}) \\ &= \sum_{i=1}^{m} \log P(y_i|\mathbf{x}i; \boldsymbol{\theta}) \\ &= \sum_{i=1}^{m} \log \left[(h_\theta(\mathbf{x}i))^{y_i}(1 - h\theta(\mathbf{x}i))^{1-y_i}\right] \\ &= \sum_{i=1}^{m} [y_i \log(h_\theta(\mathbf{x}i)) + (1 - y_i)\log(1 - h\theta(\mathbf{x}_i))] \end{aligned}$$

To maximize the log-likelihood, we can use gradient ascent. The update rule for a parameter θ_j is:

$$\theta_j := \theta_j + \alpha \frac{\partial}{\partial \theta_j} l(\boldsymbol{\theta}) \tag{15}$$

where α is the learning rate.

Let's find the partial derivative with respect to θ_j. We'll focus on a single training example (x, y) first and use the chain rule. Let $h = h_\theta(\mathbf{x}) = g(\boldsymbol{\theta}^T\mathbf{x})$, where $g(z) = \frac{1}{1+e^{-z}}$ is the sigmoid function.

The derivative of the sigmoid function $g(z)$ with respect to z is:

$$g'(z) = g(z)(1 - g(z)) \tag{16}$$

Now, let's compute the partial derivative of the log-likelihood contribution from a single example:

$$\begin{aligned} \frac{\partial}{\partial \theta_j}[y \log(h) + (1 - y)\log(1 - h)] &= \left(\frac{y}{h} - \frac{1-y}{1-h}\right)\frac{\partial h}{\partial \theta_j} \\ &= \left(\frac{y(1-h) - (1-y)h}{h(1-h)}\right)\frac{\partial h}{\partial \theta_j} \\ &= \left(\frac{y - yh - h + yh}{h(1-h)}\right)\frac{\partial h}{\partial \theta_j} \\ &= \left(\frac{y - h}{h(1-h)}\right)\frac{\partial h}{\partial \theta_j} \end{aligned}$$

Next, we compute $\frac{\partial h}{\partial \theta_j}$ using the chain rule again:

$$\frac{\partial h}{\partial \theta_j} = \frac{\partial g(\boldsymbol{\theta}^T \mathbf{x})}{\partial \theta_j} = g'(\boldsymbol{\theta}^T \mathbf{x}) \cdot \frac{\partial (\boldsymbol{\theta}^T \mathbf{x})}{\partial \theta_j} = h(1-h) \cdot x_j \tag{17}$$

Substituting this back into our expression:

$$\frac{\partial}{\partial \theta_j} l(\boldsymbol{\theta}) = \left(\frac{y-h}{h(1-h)} \right) \cdot (h(1-h) \cdot x_j) \qquad = (y-h)x_j \;= (y_i - h_\theta(\mathbf{x}_i))x_i^{(j)}$$

To get the gradient for the entire training set, we sum over all m examples:

$$\frac{\partial}{\partial \theta_j} l(\boldsymbol{\theta}) = \sum_{i=1}^{m} (y_i - h_\theta(\mathbf{x}_i))x_i^{(j)} \tag{18}$$

Finally, substituting this into the gradient ascent update rule gives us:

$$\theta_j := \theta_j + \alpha \sum_{i=1}^{m} (y_i - h_\theta(\mathbf{x}_i))x_i^{(j)} \tag{19}$$

This update rule is identical in form to the one for linear regression, but note that the definition of $h_\theta(\mathbf{x})$ is different.

4 Regression with Non-linear Features

While standard regression models are designed to find linear relationships between variables, many real-world datasets exhibit patterns that are curved or more complex. We can capture these relationships by transforming our input features into a non-linear space before fitting a linear model.

4.1 Basis Function Expansion

The core idea is to apply a non-linear transformation $\phi(x)$ to our original features. Instead of predicting the output directly from x, we use a set of **basis functions**. For example, in **Polynomial Regression**, we treat powers of x as independent features:

$$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots + \theta_d x^d + \epsilon$$

Even though the relationship between x and y is a curve, the model remains **linear in the parameters** (θ). This means we can still use efficient linear optimization techniques to find the best fit.

4.2 The Linearity of Logistic Regression

It is a common misconception that **Logistic Regression** is a non-linear model because it outputs probabilities via the S-shaped sigmoid function. However, Logistic Regression is fundamentally a **linear classifier**.

Because the decision boundary—where the model decides between two classes—is defined by the equation $\theta^T x = 0$, it forms a straight line (or hyperplane) in the feature space. To capture non-linear decision boundaries, one must manually introduce non-linear features, such as interaction terms ($x_i x_j$) or polynomial terms (x_i^2).

4.3 Common Transformation Techniques

- **Polynomial Features:** Adding $x^2, x^3, \ldots$ to capture parabolic or cubic trends.
- **Interaction Terms:** Adding $x_1 \cdot x_2$ to model synergies between features.
- **Logarithmic Transforms:** Used when data spans several orders of magnitude.
- **Splines:** Using piecewise polynomials to fit different regions of the data more flexibly.

5 The Bias-Variance Trade-off

This part explores the concepts of overfitting and underfitting from a mathematical perspective. We will use a regression model to formalize these ideas and introduce the bias-variance decomposition, a fundamental tool for understanding model performance.

5.1 Regression with Gaussian Noise

Let's consider a regression problem where the true underlying relationship between an input $\mathbf{x}$ and an output y is given by a function $h(x)$, but the observed data is corrupted by noise.

5.1.1 Model Definition

We model the observed output y as:

$$y = h(x) + \epsilon$$

Where:

- $h(x)$ is the true, unknown function we want to approximate.
- ϵ is a random noise term, which we assume is normally distributed with a mean of zero and a variance of σ^2, i.e., $\epsilon \sim \mathcal{N}(0, \sigma^2)$.

5.1.2 Training Data and Regressor

Our training data, $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, is a set of samples drawn from some joint probability distribution $P(x, y)$. We assume these samples are independent and identically distributed (i.i.d.).

Given a specific training set D, we can train a regressor (or hypothesis) which we denote as $h_D(x)$. The subscript D emphasizes that the learned model depends on the specific training data it was given.

5.2 Decomposing the Expected Error

Our goal is to understand the performance of our learning algorithm not just on one particular training set, but on average. We want to measure the expected error of our trained regressor $h_D(x)$ on new, unseen data points (x, y).

The squared error for a single prediction is $(h_D(x) - y)^2$.

5.2.1 Expected Error

The expected error is taken over all possible training sets D of a given size n, and all possible new data points (x, y). This gives us a measure of the expected quality of our model.

The full expression for the expected test error is:

$$E_{\text{total}} = E_{(x,y)\sim P, D\sim P^n}\left[(h_D(x) - y)^2\right] = \int_D \int_x \int_y (h_D(x) - y)^2 P(x, y) P(D)\, dx\, dy\, dD$$

To understand the sources of this error, we can decompose this expression. Let's define two key quantities:

1. **The average regressor**, $\bar{h}(x)$: This is the expected hypothesis if we could average over all possible training sets D.

$$\bar{h}(x) \triangleq E_D[h_D(x)] = \int h_D(x) P(D)\, dD$$

2. **The expected groundtruth**, $\bar{y}(x)$: This is the expected value of the target y for a given input x.

$$\bar{y}(x) \triangleq E_{y|x}[y] = \int y P(y|x)\, dy$$

With some algebraic manipulation (specifically, adding and subtracting $\bar{h}(x)$ and $\bar{y}(x)$ inside the squared term), the total expected error can be decomposed into three components:

$$E_{\text{total}} = \underbrace{E_{x,D}\left[(h_D(x) - \bar{h}(x))^2\right]}_{\text{Variance}} + \underbrace{E_x\left[(\bar{h}(x) - \bar{y}(x))^2\right]}_{\text{Bias}^2} + \underbrace{E_{x,y}\left[(y - \bar{y}(x))^2\right]}_{\text{Noise}}$$

Let's analyze each term.

5.2.2 Variance

$$\text{Variance} = E_{x,D}\left[(h_D(x) - \bar{h}(x))^2\right]$$

Variance measures how much our learned model $h_D(x)$ changes as the training data D changes. It captures the model's sensitivity to the specific training set. A high variance means the model can change drastically with small changes in the training data, which is a hallmark of **overfitting**. Such a model fits the training data's noise rather than its underlying signal.

5.2.3 Bias

$$\text{Bias}^2 = E_x\left[(\bar{h}(x) - \bar{y}(x))^2\right]$$

Bias measures the inherent error of our learning algorithm. It is the squared difference between the average regressor $\bar{h}(x)$ (what our model learns on average) and the true expected groundtruth $\bar{y}(x)$. A high bias means that even with infinite training data, our model is fundamentally incapable of representing the true relationship. This is a sign of **underfitting**. For example, using a linear model to fit a non-linear phenomenon will result in high bias.

5.2.4 Noise

$$\text{Noise} = E_{x,y}\left[(y - \bar{y}(x))^2\right]$$

Noise, also known as irreducible error, is the error that arises from the data itself. It's a measure of the inherent randomness or ambiguity in the problem. This error cannot be reduced by choosing a better model; it is a property of the data distribution.

5.3 The Trade-off and Model Complexity

The relationship between bias and variance leads to a fundamental dilemma in machine learning:

- As we increase model complexity (e.g., using a higher-degree polynomial), the model becomes more flexible. This generally **decreases bias** (as it can fit more complex patterns) but **increases variance** (as it is more likely to overfit to the training data).
- Conversely, simplifying a model **increases bias** but **decreases variance**.

This is known as the **bias-variance trade-off**. The goal is to find an optimal model complexity that minimizes the total error, which is the sum of bias-squared and variance.

6 Summary of Linear Regression

The concepts discussed are general, but they are often introduced in the context of linear regression. Here's a quick summary of key linear regression formulas:

- **Goal:** Minimize a loss function, typically the Mean Squared Error (MSE).
- **Direct Solution (Normal Equation):** For linear regression, there is a closed-form solution.

$$\theta = (X^T X)^{-1} X^T y$$

- **Iterative Solution (Gradient Descent):** The parameters can also be updated iteratively. The update rule for a parameter θ_j is:

$$\theta_j \leftarrow \theta_j + r \sum_{i=1}^{m} (y_i - h_\theta(x_i)) x_i^{(j)}$$

where r is the learning rate, m is the number of training examples, and $h_\theta(x_i)$ is the model's prediction for the i-th example. For linear regression, $h_\theta(x_i) = \theta^T x_i$.

For linear regression, the problem is convex, meaning both the direct and iterative methods should converge to the same optimal solution, provided the learning rate is appropriately chosen for gradient descent.

7 Bayesian Decision Theory

Bayesian decision theory provides a probabilistic framework for classification. The goal is to make a classification guess based on probability distributions. We can use either:

- $P(x|\omega_j)$: The class-conditional probability density (how likely is feature x given class ω_j).
- $P(\omega_j|x)$: The posterior probability (how likely is class ω_j given feature x).

7.1 Bayes' Theorem

The two probabilities are related by Bayes' theorem:

$$P(\omega_j|x) = \frac{P(x|\omega_j)P(\omega_j)}{P(x)}$$

Where:

- $P(\omega_j|x)$ is the **posterior** probability.
- $P(x|\omega_j)$ is the **likelihood**.
- $P(\omega_j)$ is the **prior** probability of the class.
- $P(x)$ is the **evidence** (marginal probability of x).

7.2 Frequentist vs. Bayesian View

There are two main philosophical approaches to interpreting probability and statistical inference: the Frequentist view and the Bayesian view.

7.2.1 Frequentist View

- **Definition of Probability**: Probability is defined as the "frequency of occurrence" of an event in the long run.
- **Nature of Data**: Data is viewed as the result of a random procedure (sampling) drawn from an unknown but fixed phenomenon.
- **Nature of Parameters**: The distribution parameter (θ) is considered an unknown but **fixed constant**. It does not have a probability distribution itself.

7.2.2 Bayesian View

- **Definition of Probability**: Probability represents a "degree of uncertainty" or belief about an event or hypothesis.
- **Nature of Data**: The observed data is treated as **fixed**. The goal is to infer the properties of the unknown phenomenon given this specific data.
- **Nature of Parameters**: The distribution parameter (θ) is treated as a random variable with its own probability distribution.
- **Role of Prior Knowledge**: Prior knowledge or beliefs about the phenomenon are explicitly modeled and can influence the final inference results.

8 Parameter Estimation

To use these probabilistic models, we first need to estimate the parameters of the distributions from data.

8.1 Maximum Likelihood Estimation (MLE)

The MLE approach finds the parameter θ that maximizes the likelihood of observing the given data.

- **Viewpoint**: Frequentist. Assumes θ is a fixed, unknown constant.
- **Objective**: Maximize the likelihood function, $L(\theta) = P(X|\theta)$.

$$\hat{\theta}_{MLE} = \arg\max_{\theta} P(X|\theta)$$

- **Method**: Often done by maximizing the log-likelihood, which simplifies the math (products become sums). The solution is found by taking the derivative with respect to θ, setting it to zero, and solving for θ.

Example: MLE for Gaussian Mean For a set of IID observations $\{x_1, \ldots, x_n\}$ from a Gaussian distribution with unknown mean μ and known variance σ^2, the MLE of the mean is the sample mean:

$$\hat{\mu}_{MLE} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

8.2 Maximum A Posteriori (MAP) Estimation

The MAP approach finds the most probable parameter θ given the observed data and a prior belief about θ.

- **Viewpoint**: Bayesian. Assumes θ is a random variable with a prior distribution $P(\theta)$.
- **Objective**: Maximize the posterior probability of the parameters.

$$\hat{\theta}_{MAP} = \arg\max_{\theta} P(\theta|X) = \arg\max_{\theta} \frac{P(X|\theta)P(\theta)}{P(X)}$$

  Since $P(X)$ is constant with respect to θ, this simplifies to:

$$\hat{\theta}_{MAP} = \arg\max_{\theta} P(X|\theta)P(\theta)$$

- **Notes**:
  - If the prior $P(\theta)$ is a uniform (uninformative) distribution, MAP reduces to MLE.
  - As the amount of data increases, the MAP estimate converges to the MLE estimate.
  - Priors help to regularize the model and prevent overfitting, especially with small datasets.

8.3 Comparison: MLE vs. MAP

The following table summarizes the key differences between Maximum Likelihood Estimation (MLE) and Maximum A Posteriori (MAP) estimation.

Feature	**Maximum Likelihood (MLE)**	**Maximum A Posteriori (MAP)**
Viewpoint	Frequentist	Bayesian
Assumption on θ	θ is a fixed, unknown constant.	θ is a random variable with a specific distribution.
Maximization Objective	Likelihood function: $L(\theta) = P(X\|\theta)$	Posterior probability: $P(\theta\|X) \propto P(X\|\theta)P(\theta)$
Role of Prior	No prior is used (data determines everything).	Uses a prior $P(\theta)$ to influence the estimate.
Small Data Behavior	Prone to overfitting (e.g., seeing 3 heads in 3 tosses implies $P(H) = 1$).	Regularized by the prior (prevents extreme estimates).
Relationship	Equivalent to MAP with a uniform prior.	Converges to MLE as sample size $n \to \infty$.

9 Bayesian Classifiers: Likelihood Ratio Test

The goal of a classifier is to assign a class, ω_i, to an observation vector, x. A common approach is to choose the class that is most probable given the observation.

- If $P(\omega_1|x) > P(\omega_2|x)$, then the observation x is more likely to belong to class ω_1.
- However, the posterior probability $P(\omega_i|x)$ can be difficult to compute directly. It is often more intuitive and easier to calculate the class-conditional probability (likelihood) $P(x|\omega_i)$ from a training dataset.
- Using Bayes' theorem, $P(\omega_i|x) = \frac{P(x|\omega_i)P(\omega_i)}{P(x)}$, we can rewrite the decision rule.

Our classifier compares the posterior probabilities of the two classes:

$$P(\omega_1|x) \gtrless^{\omega_1}_{\omega_2} P(\omega_2|x)$$

Using Bayes' theorem and canceling the common evidence term $P(x)$, our classifier becomes:

$$P(x|\omega_1)P(\omega_1) \gtrless^{\omega_1}_{\omega_2} P(x|\omega_2)P(\omega_2)$$

This can be rearranged into the Likelihood Ratio Test (LRT):

$$\underbrace{\frac{P(x|\omega_1)}{P(x|\omega_2)}}_{\text{Likelihood Ratio}} \gtrless^{\omega_1}_{\omega_2} \underbrace{\frac{P(\omega_2)}{P(\omega_1)}}_{\text{Ratio of Priors}}$$

We decide for class ω_1 if the likelihood ratio is greater than the ratio of prior probabilities.

10 Notes on the Likelihood Ratio Test (LRT)

10.1 Minimizing Classification Error

The Likelihood Ratio Test, as defined above, minimizes the total probability of classification error, assuming that all types of errors are equally costly. This is also known as the **MAP (Maximum A Posteriori) decision rule**. The classifier is often called the **Bayes classifier**.

The decision boundary is set at the point where $P(x|\omega_1)P(\omega_1) = P(x|\omega_2)P(\omega_2)$. The total error is the sum of the probabilities of misclassifying an instance from class ω_1 as ω_2 and vice-versa.

10.2 Bayes Loss/Risk Classifier

In many real-world scenarios, not all errors are equal. For example, misdiagnosing a sick patient as healthy might be far more costly than misdiagnosing a healthy patient as sick. We can account for this by assigning a loss or risk, $L_{i|j}$, to the action of deciding class ω_i when the true class is ω_j.

To minimize the total expected loss, the decision rule is modified. This is called the **Bayes loss/risk classifier**. The new decision rule is:

$$\frac{P(x|\omega_1)}{P(x|\omega_2)} \gtrless_{\omega_2}^{\omega_1} \frac{P(\omega_2)(L_{1|2} - L_{2|2})}{P(\omega_1)(L_{2|1} - L_{1|1})}$$

- $L_{1|2}$ is the loss for deciding ω_1 when the true class is ω_2.
- $L_{2|1}$ is the loss for deciding ω_2 when the true class is ω_1.
- $L_{1|1}$ and $L_{2|2}$ are the losses for correct classifications, which are typically zero.

Zero-One Loss When we treat all errors equally, we use a **zero-one loss** function. Here, the loss is 1 for any misclassification and 0 for any correct classification.

- $L_{1|2} = 1$ (misclassification)
- $L_{2|1} = 1$ (misclassification)
- $L_{1|1} = 0$ (correct classification)
- $L_{2|2} = 0$ (correct classification)

Substituting these values into the Bayes risk formula simplifies it back to the original LRT:

$$\frac{P(x|\omega_1)}{P(x|\omega_2)} \gtrless_{\omega_2}^{\omega_1} \frac{P(\omega_2)(1-0)}{P(\omega_1)(1-0)} = \frac{P(\omega_2)}{P(\omega_1)}$$

10.3 Maximum Likelihood Criterion

If we have no reason to believe one class is more probable than another, we can assume equal priors, i.e., $P(\omega_1) = P(\omega_2)$. In this case, the ratio of priors becomes 1.

This simplifies the LRT to the **Maximum Likelihood (ML) criterion**:

$$\frac{P(x|\omega_1)}{P(x|\omega_2)} \gtrless_{\omega_2}^{\omega_1} 1$$

Or, more simply:

$$P(x|\omega_1) \gtrless_{\omega_2}^{\omega_1} P(x|\omega_2)$$

In this case, we choose the class for which the observed data is most likely.

10.4 Likelihood Ratio Test (LRT)

The optimal decision rule under a zero-one loss function (where all errors are equally costly) is to choose the class with the highest posterior probability. For a two-class problem, we decide ω_1 if:

$$P(\omega_1|x) > P(\omega_2|x)$$

Using Bayes' theorem, this is equivalent to the Likelihood Ratio Test:

$$\frac{P(x|\omega_1)}{P(x|\omega_2)} > \frac{P(\omega_2)}{P(\omega_1)}$$

This test, also known as the **Bayes Classifier**, minimizes the total classification error.

11 Naïve Bayes Classifier

The Curse of Dimensionality For a feature vector $x = [x_1, \dots, x_d]$, the likelihood $P(x|\omega_j)$ is a d-dimensional distribution. Modeling this directly is difficult and requires an enormous amount of data to avoid sparsity (many possible feature combinations will not appear in the training set).

The "Naïve" Assumption The Naïve Bayes classifier simplifies this problem by making a strong assumption of **conditional independence**. It assumes that all features are independent of each other, given the class:

$$P(x|\omega_j) = \prod_{k=1}^{d} P(x_k|\omega_j)$$

This simplifies the decision rule. We now classify by choosing the class ω_j that maximizes:

$$\arg\max_{\omega_j} P(\omega_j) \prod_{k=1}^{d} P(x_k|\omega_j)$$

This requires estimating only d one-dimensional distributions $P(x_k|\omega_j)$ for each class, which is much more feasible.

Handling Missing Data Oftentimes, we have missing values in our features. In such cases, many models require us to perform data imputation.[1] However, for Naïve Bayes, missing values are handled naturally by simply **omitting the missing attribute** from the likelihood calculation for a given instance. Because the model assumes features are conditionally independent, the joint probability can be calculated **using only the available features** without bias. During the training phase, the model calculates the probability of the available features given the class, and during prediction, it ignores any features that are unknown, effectively treating them as if they provide no information for that specific classification.

Dealing with Zero Probabilities A practical issue arises if a feature value x_k never appears with a class ω_j in the training data. This would make $P(x_k|\omega_j) = 0$, causing the entire posterior product to become zero. To handle this, we use smoothing techniques, such as:

- **Laplace (Additive) Smoothing**: Add a small constant (e.g., 1) to all counts, ensuring no probability is ever zero.
- **Using Priors (MAP adaptation)** to inform the estimates.

Despite its strong assumption, the Naïve Bayes classifier is remarkably effective in practice, especially in text classification and other NLP tasks. It is fast, handles missing data well, and provides a good baseline.

[1] https://scikit-learn.org/stable/modules/impute.html