Random Forest and Deep Learning

Carlos E Martínez-Rodríguez

14 de noviembre de 2023

Índice

1. Bioinformatics: New revision

1.1. A bayesian framework for combining gene predictions - Pavlovic

Biology and biotechnology are undergoing a techonolgical revolution which is transforming research into an information-rich enterprise. A typical bacterial genome sequence is comprised of several million bases of DNA and contains several thousand genes. The human genome is approximately 3 billion bases long and it contains approximately 30,000 putative genes identified thus far.

2. Random Forest: Explanation in Mathematical Terms

Random Forest is a supervised learning algorithm used for both classification and regression problems. The main idea behind Random Forest is to build multiple decision trees during training and combine their results to obtain a more robust and accurate prediction.

Let's assume we have a training dataset $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, where x_i represents the features and y_i is the target variable.

1. **Construction of Decision Trees:**

   - For each tree, a random subset of features is selected (randomly sample features), and a random subset of training data is chosen (randomly sample observations with replacement).
   - A decision tree is constructed using the subset of data and features.
   - The decision tree is represented as $h_i(x; \theta_i)$, where i denotes the tree index, x is the input feature vector, and θ_i represents the parameters of the tree.
   - At each node t of the tree, a feature j_t is selected from the random subset, and the split is determined based on minimizing impurity:

     $$\theta_{i,t} = \arg\min_{j_t, s_t} [\text{Impurity}(D_t) - p_{\text{left}}\text{Impurity}(D_{\text{left}}) - p_{\text{right}}\text{Impurity}(D_{\text{right}})]$$

     where D_t is the dataset at node t, D_{left} and D_{right} are the datasets in the left and right child nodes, p_{left} and p_{right} are the proportions of data in the left and right child nodes, and $\text{Impurity}(\cdot)$ is a measure of impurity, such as Gini index for classification or mean squared error for regression.

2. **Voting or Averaging:**

   - For classification problems, the final prediction is obtained by majority voting among the trees:

     $$H(x) = \text{mode}\{h_1(x; \theta_1), h_2(x; \theta_2), \ldots, h_n(x; \theta_n)\}$$

- For regression problems, the final prediction is the average of predictions from all trees:

$$H(x) = \frac{1}{n} \sum_{i=1}^{n} h_i(x; \theta_i)$$

3. **Variance Reduction:**

   - By building decision trees in a random manner, the variance of the model is reduced, leading to a more robust and generalizable model.

The final prediction is obtained by combining the predictions of all trees.

3. Deep Learning: Overview

Deep Learning is a subfield of machine learning that focuses on neural networks with multiple layers (deep neural networks). These networks can automatically learn hierarchical representations of data, allowing them to capture intricate patterns and features.

1. **Neural Network Representation:**

   - A neural network consists of layers of interconnected nodes (neurons) organized into an input layer, one or more hidden layers, and an output layer.
   - The input layer represents the features of the data, and each neuron in the layer processes a specific feature.
   - Hidden layers perform nonlinear transformations on the input data, learning hierarchical representations.
   - The output layer produces the final prediction based on the learned representations.

2. **Training a Neural Network:**

   - During training, the network's parameters (weights and biases) are adjusted to minimize the difference between predicted and actual outputs.
   - This optimization is typically done using backpropagation and gradient descent.

3. **Activation Functions:**

   - Activation functions introduce nonlinearity to the neural network, enabling it to learn complex patterns.
   - Common activation functions include ReLU (Rectified Linear Unit), Sigmoid, and Hyperbolic Tangent (tanh).

4. **Forward Pass:**

   - The forward pass of a neural network involves computing the output of the network for a given input.

- Given an input vector x, the output y is computed by passing x through the network's layers using learned weights and biases.
- The output of each layer is computed as:

$$a^{(l)} = W^{(l)}a^{(l-1)} + b^{(l)}$$

  where $W^{(l)}$ is the weight matrix, $a^{(l)}$ is the activation of layer l, $a^{(l-1)}$ is the activation of the previous layer, and $b^{(l)}$ is the bias vector.
- The activation function is then applied to $a^{(l)}$ to introduce nonlinearity.

5. **Loss Function:**
   - The loss function measures the difference between the predicted output and the true output.
   - Common loss functions include mean squared error for regression and cross-entropy for classification.
   - The goal during training is to minimize the loss by adjusting the network's parameters.
6. **Backpropagation:**
   - Backpropagation is a training algorithm that computes the gradient of the loss with respect to the network's parameters.
   - The gradient is used to update the parameters in the direction that reduces the loss.
   - It involves computing the gradient of the loss with respect to the output of each layer and using the chain rule to propagate these gradients backward through the network.

4. Example of Implementation in R:

4.1. Install and Load the Keras Library:

```
install.packages("keras")
library(keras)
```

4.2. Build a Simple Neural Network:

```
model <- keras_model_sequential() %>%
  layer_dense(units = 128, activation = 'relu', input_shape = c(10)) %>%
  layer_dense(units = 1, activation = 'sigmoid')

summary(model)
```

4.3. Compile and Train the Neural Network:

```
model %>% compile(
  optimizer = 'adam',
  loss = 'binary_crossentropy',
  metrics = c('accuracy')
)

# Assuming you have training data X_train and labels y_train
history <- model %>% fit(
  X_train, y_train,
  epochs = 10, batch_size = 32,
  validation_split = 0.2
)
```

4.4. Evaluate the Model:

```
# Assuming you have test data X_test and labels y_test
evaluate_result <- model %>% evaluate(X_test, y_test)
print(evaluate_result)
```

5. Impurity Measures in Random Forest

In the context of Random Forest, impurity measures play a crucial role in the construction of decision trees. The two commonly used impurity measures are the Gini index for classification and the mean squared error for regression.

5.1. Gini Index for Classification

The Gini index is a measure of impurity used in classification problems. Given a node in a decision tree that contains data points from different classes, the Gini index quantifies how often a randomly chosen data point would be incorrectly classified.

For a node t with K classes and a set of data points D_t, the Gini index ($Gini(t)$) is calculated as follows:

$$Gini(t) = 1 - \sum_{i=1}^{K} p_i^2$$

where p_i is the proportion of data points in class i at node t. A lower Gini index indicates a purer node with predominantly one class.

In the context of Random Forest, the decision tree split is determined by minimizing the weighted sum of Gini indices for the left and right child nodes. The split that results in the lowest overall Gini index is chosen.

5.2. Mean Squared Error for Regression

For regression problems, the impurity measure used is the mean squared error (MSE). Unlike classification, where impurity is related to the purity of classes in a node, regression impurity is a measure of the variability of target values within a node.

For a node t with data points D_t, the MSE ($MSE(t)$) is calculated as follows:

$$MSE(t) = \frac{1}{|D_t|} \sum_{i \in D_t} (y_i - \bar{y}_t)^2$$

where y_i is the target value of data point i, $|D_t|$ is the number of data points in node t, and $\bar{y}_t$ is the mean target value of all data points in node t.

Similar to the Gini index, in Random Forest, the decision tree split is determined by minimizing the weighted sum of MSE for the left and right child nodes.

These impurity measures guide the construction of individual decision trees within the Random Forest ensemble, contributing to the overall robustness and predictive power of the model.

6. Construcción del Árbol de Decisión

6.1. Teoría:

Un árbol de decisión es una estructura de datos que representa un conjunto de decisiones y sus posibles consecuencias. En el contexto de Random Forest, la construcción de un árbol de decisión sigue el principio de .ªprendizaje supervisado", donde el algoritmo aprende patrones a partir de un conjunto de datos etiquetado.

La construcción del árbol se realiza a través de divisiones recursivas basadas en características del conjunto de datos. Cada nodo del árbol representa una pregunta sobre una característica, y las ramas que surgen de ese nodo son las respuestas a esa pregunta. El proceso continúa hasta que se alcanza un criterio de parada, como la profundidad máxima del árbol o el número mínimo de muestras en un nodo.

6.2. Elementos Matemáticos:

1. **Función de Impureza:**

En cada nodo del árbol, se elige la característica y el umbral que minimizan la impureza en los nodos hijos resultantes. La impureza se mide mediante funciones como el Índice de Gini para clasificación o el Error Cuadrático Medio (MSE) para regresión.

Para clasificación:

$$Gini(t) = 1 - \sum_{i=1}^{K} p_i^2$$

Donde p_i es la proporción de ejemplos de la clase i en el nodo t.

Para regresión:

$$MSE(t) = \frac{1}{|D_t|} \sum_{i \in D_t} (y_i - \bar{y}_t)^2$$

Donde D_t es el conjunto de datos en el nodo t, y_i es la etiqueta del ejemplo i, y $\bar{y}_t$ es la media de las etiquetas en el nodo t.

2. **Criterio de División:**

La elección de la mejor característica y umbral se basa en la reducción de la impureza. Se busca el par (j, s) que minimiza la expresión:

$$\theta_{i,t} = \arg\min_{j,s} [\text{Impureza}(D_t) - p_{\text{left}}\text{Impureza}(D_{\text{left}}) - p_{\text{right}}\text{Impureza}(D_{\text{right}})]$$

Donde D_t es el conjunto de datos en el nodo t, D_{left} y D_{right} son los conjuntos de datos en los nodos izquierdo y derecho después de la división, y p_{left} y p_{right} son las proporciones de datos en esos nodos.

3. **Criterios de Parada:**

Para evitar sobreajuste, se utilizan criterios de parada, como la profundidad máxima del árbol o el número mínimo de muestras requeridas para realizar una división.

La construcción de cada árbol en Random Forest implica este proceso iterativo y se repite para cada árbol en el bosque. La diversidad en la construcción de árboles se logra mediante el uso de diferentes subconjuntos aleatorios de características y datos en cada árbol. La combinación de predicciones de estos árboles mejora la generalización del modelo y su capacidad predictiva en nuevos datos.

7. Reducción de Varianza en Random Forest

7.1. Teoría:

La reducción de la varianza es uno de los objetivos clave en la construcción de árboles de decisión en Random Forest. Esta técnica busca mejorar la generalización del modelo al reducir la sensibilidad a pequeñas variaciones en los datos de entrenamiento.

La varianza se refiere a la variabilidad de las predicciones de un modelo respecto a diferentes conjuntos de datos de entrenamiento. En Random Forest, la reducción de la varianza se logra mediante dos técnicas principales: Bagging y Random Subspace.

7.1.1. Bagging (Bootstrap Aggregating):

Bagging es una técnica que consiste en entrenar múltiples modelos en diferentes subconjuntos de datos generados mediante muestreo con reemplazo (bootstrap). Cada árbol de decisión en Random Forest se entrena en un conjunto de datos ligeramente diferente, lo que introduce diversidad en los modelos.

7.1.2. Random Subspace:

Random Subspace es otra técnica utilizada para reducir la correlación entre los árboles de decisión. En lugar de usar todas las características para cada árbol, se selecciona un subconjunto aleatorio de características para entrenar cada árbol. Esto ayuda a que cada árbol se especialice en diferentes aspectos de los datos, mejorando la diversidad y reduciendo la correlación entre las predicciones.

7.2. Elementos Matemáticos:

La reducción de la varianza no se expresa directamente con fórmulas matemáticas específicas, pero los conceptos clave son fundamentales:

1. **Bagging:** - Cada árbol T_i se entrena en un conjunto de datos D_i generado por muestreo con reemplazo (bootstrap). - La predicción final se obtiene promediando las predicciones de todos los árboles:

$$H(x) = \frac{1}{N} \sum_{i=1}^{N} T_i(x)$$

2. **Random Subspace:** - Cada árbol T_i se entrena utilizando un subconjunto aleatorio de características. - La predicción final se obtiene promediando las predicciones de todos los árboles:

$$H(x) = \frac{1}{N} \sum_{i=1}^{N} T_i(x)$$

Estos enfoques combinados contribuyen a reducir la varianza del modelo, lo que resulta en un modelo más robusto y generalizable.

8. Votación y Promedio en Random Forest

8.1. Teoría:

La técnica de "Votación"(para problemas de clasificación) o "Promedio"(para problemas de regresión) es crucial en Random Forest para combinar las predicciones de múltiples árboles de decisión y obtener una predicción final más robusta y precisa.

8.1.1. Para Problemas de Clasificación:

En problemas de clasificación, el enfoque de votación se utiliza. Cada árbol en el bosque emite una predicción de clase, y la clase final se determina por mayoría de votos.

8.1.2. Para Problemas de Regresión:

En problemas de regresión, se utiliza un enfoque de promedio. Cada árbol realiza una predicción numérica, y la predicción final es el promedio de todas las predicciones.

8.2. Elementos Matemáticos:

1. **Votación para Clasificación:** - La predicción final para un ejemplo x se obtiene por mayoría de votos:

$$H(x) = \text{mode}\{T_1(x), T_2(x), ..., T_N(x)\}$$

2. **Promedio para Regresión:** - La predicción final para un ejemplo x se obtiene promediando las predicciones de todos los árboles:

$$H(x) = \frac{1}{N} \sum_{i=1}^{N} T_i(x)$$

Donde $T_i(x)$ representa la predicción del árbol i para el ejemplo x, y N es el número total de árboles en el bosque.

Estos enfoques de votación y promedio permiten que Random Forest combine la información de múltiples árboles de decisión, mejorando la generalización y la capacidad predictiva del modelo.

9. Resumen

Random Forest es un algoritmo de aprendizaje automático que se utiliza para resolver problemas de clasificación y regresión. En lugar de utilizar un único árbol de decisión, Random Forest construye varios árboles de decisión y los combina para obtener una predicción más precisa. Cada árbol de decisión se construye utilizando un subconjunto aleatorio de las características del conjunto de datos original. Al final, las predicciones de cada árbol se promedian para obtener una predicción final.

El algoritmo de Random Forest se basa en dos técnicas: Bagging y Random Subspace. Bagging es una técnica que se utiliza para reducir la varianza de un modelo al entrenar múltiples modelos en diferentes subconjuntos de datos. Random Subspace es una técnica que se utiliza para reducir la correlación entre los modelos al entrenar cada modelo en diferentes subconjuntos de características.

Aquí hay algunos elementos matemáticos que se utilizan en Random Forest:

- **Árbol de decisión:** Un árbol de decisión es una estructura de datos que se utiliza para modelar decisiones y sus posibles consecuencias. Cada nodo en el árbol representa una decisión, y cada rama representa una posible consecuencia de esa decisión. Los árboles de decisión se construyen utilizando un conjunto de reglas que se utilizan para tomar decisiones.

- **Bootstrap:** Bootstrap es una técnica que se utiliza para generar múltiples conjuntos de datos a partir de un conjunto de datos original. Cada conjunto de datos se genera mediante muestreo con reemplazo, lo que significa que cada elemento del conjunto de datos original tiene la misma probabilidad de ser seleccionado en cada conjunto de datos generado.

- **Out-of-Bag Error:** Out-of-Bag Error es una técnica que se utiliza para estimar el error de validación de un modelo sin la necesidad de un conjunto de datos de validación separado. El error se estima utilizando los datos que no se incluyeron en el conjunto de datos de entrenamiento para cada árbol de decisión.

9.1. Ejemplo en R: Random Forest

Aquí hay un ejemplo de cómo construir un modelo de bosque aleatorio en R:

```
# Cargamos el paquete necesario para este ejemplo
library(randomForest)

# Cargamos el conjunto de datos que deseamos utilizar
data(iris)

# Dividimos el conjunto de datos en conjuntos de entrenamiento y prueba
trainIndex <- createDataPartition(iris$Species, p = .8, list = FALSE, times = 1)

# Entrenamos el modelo de bosque aleatorio
rf_model <- randomForest(Species ~ ., data = iris[trainIndex,])

# Realizamos predicciones en el conjunto de prueba
predictions <- predict(rf_model, iris[-trainIndex,])
```

9.2. Deep Learning

Deep Learning es un subcampo del aprendizaje automático que se centra en la creación de redes neuronales artificiales profundas. Estas redes neuronales están diseñadas para imitar el cerebro humano y son capaces de aprender patrones complejos en los datos.

Aquí hay un ejemplo de cómo construir una red neuronal profunda en R utilizando el paquete keras:

```
# Cargamos los paquetes necesarios
library(keras)
library(tensorflow)

# Cargamos el conjunto de datos que deseamos utilizar
data(iris)

# Dividimos el conjunto de datos en conjuntos de entrenamiento y prueba
trainIndex <- createDataPartition(iris$Species, p = .8, list = FALSE, times = 1)

# Creamos la red neuronal
model <- keras_model_sequential() %>%
  layer_dense(units = 4, input_shape = c(4)) %>%
  layer_activation("relu") %>%
  layer_dense(units = 3) %>%
  layer_activation("softmax")

# Compilamos la red neuronal
model %>% compile(loss = "categorical_crossentropy", optimizer = "adam", metrics = "accu
```

```
# Entrenamos la red neuronal
history <- model %>% fit(
  x = iris[trainIndex, 1:4],
  y = to_categorical(as.numeric(iris[trainIndex, 5])),
  epochs = 100,
  batch_size = 10,
  validation_split = 0.2
)
```

10. Análisis de Componentes Principales

10.1. Introduction to Principal Components Analysis - Kristin L. Sainani

Principal components analysis (PCA) is a powerful statistical tool that can help researchers analyze datasets with many highly related predictors. PCA is a data reduction technique -that is, it reduces a larger set of predictor variables to a smaller set with minimal loss of information. PCA may be applied before running regression analyses or for exploratory purposes to help researchers understand relationships among their variables or discover patterns in their data. Besides compressing the data, PCA analysis also reveals clusters of variables that are highly related, which can give investigators a deeper understanding of their data

Nota 1. *To illustrate the value of PCA, I will apply the technique to some data pertaining to 91 female adolescent runners. These data come from a larger real dataset [1] but have been modified for use in classroom examples, and thus the results presented here are only for teaching purposes. The researchers measured a large number of potential predictors of bone density and stress fractures, including variables relating to body size and composition, training, per- formance, menstrual function, diet, and eating behaviors. An examination of the correlation coefficients between these 11 variables shows that many are highly related, even across different groups of variables. The application of PCA to these data before performing regression analyses has several benefits. The researchers have collected multiple measurements that reflect the same construct—for example, running performance or menstrual function. If they enter all of these variables into a regression model, they increase the risks of overfitting and type I errors (chance findings). However, if they ignore or discard variables, they lose valuable infor- mation. PCA analysis instead compresses the 11 original variables into a smaller subset of composite variables (called principal components) that capture most of the information in the original data. These new variables may then be used for further analyses. They have the added benefit of being uncorrelated with one another, which makes them easier to interpret and analyze. Before performing PCA analysis on the example data, I first imputed missing values, because observations that have any missing data will otherwise be omitted. I also converted the variables into standard deviation units (Z scores), because all the variables need to have the same units before PCA is applied.*

10.2. Principal component analysis - Herve Abdi and Lynne J. Williams

PCA analyzes a data table representing observations described by several dependent variables, which are, in general, inter-correlated. Its goal is to extract the important information from the data table and to express this information as a set of new orthogonal variables called principal components. PCA also represents the pattern of similarity of the observations and the variables by displaying them as points in maps

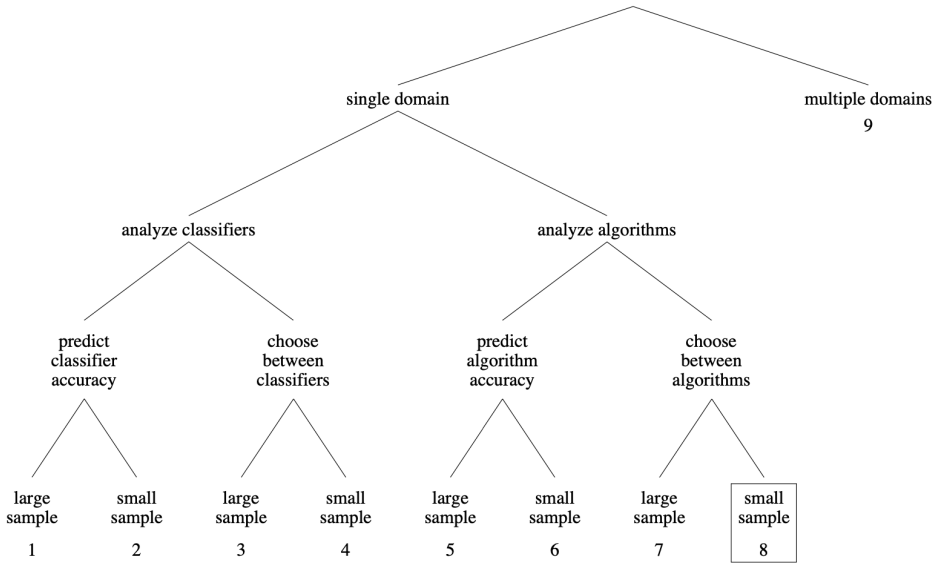
10.3. Approximate Statistical Test for comparing Supervised Classification Learning Algortihms - Dietterich Thomas

El objetivo de esta investigación es encontrar el mejor clasificador o el mejor algoritmo de aprendizaje para ser aplicado en ese dominio. Un objetivo principal en ML es encontrar algoritmos que trabajen bien en un amplio rango de aplicaciones.

Nota 2. *Un clasificador es una función que dada un ejemplo de entrada, a ese ejemplo se le asigne una de las K clases.*

Nota 3. *Un algoritmo de aprendizaje es una función que dado un conjunto de ejemplos y sus clases, construya un clasificador.*

Nota 4. *En un principio, el objetivo es encontrar el mejor clasificador y estimar su precisión en futuros ejemplos. En algunas aplicaciones se seleccionará el mejor algoritmo de aprendizaje más que el mejor clasificador.*



El siguiente nivel distingue entre estimar la precisión y elegir clasificadores o algoritmos. El nivel más bajo está relacionado con la cantidad de datos disponibles, si se tiene una gran cantidad de datos, entonces se pueden fijar un conjunto de ellos para ser considerados como datos de prueba para evaluar los clasificadores. En muchos de los casos, la cantidad de datos es limitada y se requeire utilizar todos como datos de entrada a los algoritmos de aprendizaje, esto quiere decir que se requiere utilizar algunas técnicas de remuestreo para realizar análisis estadísticos.

Nota 5. ***Nueve Preguntas***
Se supone que todos los puntos de datos se obtienen de manrea independiente de una distribución de probabilidad fija definida por el problema en particular.

Pregunta 10.1. *Supóngase que se tiene una muestra de datos grande y un clasificador C. El clasificador pudo haber sido construido utilizando parte de los datos, pero aún existen datos para conformar un conjunto de datos de prueba. Por tanto se puede medir la precisión de C en el conjunto de datos de prueba y construir un intervalo de confianza binomial. El clasificador pudo haber sido generado por cualquier método y no necesariamente por un algoritmo de aprendizaje.*

Pregunta 10.2. *Dado un conjunto de datos pequeño S, supóngase que se aplica un algoritmo de aprendizaje A a S para construir un clasificador C_A, qué tan preciso es el clasificador C_A para nuevos ejemplos. Dado que no se tienen conjuntos de datos separados no es posible responder la pregunta directamente. Es posible predecir la precisión del algoritmo A cuando se entrena en conjuntos de datos seleccionados aleatoriamente de aproximadamente el mismo tamaño que S. Si esto es posible entonces se puede predecir la precisión de C_A que fue obtenido despues de ser entrenado en S.*

Pregunta 10.3. *Dados dos clasificadores C_A y C_B y suficientes datos para tener un conjunto de datos de prueba, determinar cual clasificador será más preciso en nuevos ejemplos de prueba. Esto se puede responder midiendo la precisión de cada clasificador en los datos de prueba y aplicando la prueba de McNemar.*

Pregunta 10.4.

Pregunta 10.5.

Pregunta 10.6.

Pregunta 10.7.

Pregunta 10.8.

Pregunta 10.9.

11. Guia de estudio de Machine Learning

- Algoritmos de aprendizaje supervisado:
  - Regresión lineal y logística
  - Máquinas de soporte vectorial (SVM)
  - Árboles de decisión y bosques aleatorios
  - Redes neuronales
- Aprendizaje no supervisado:
  - K-Means y clustering jerárquico
  - Análisis de componentes principales (PCA)
  - Algoritmos de asociación
  - Mapas autoorganizados (SOM)
- Evaluación de modelos y métricas:
  - Precisión, sensibilidad, especificidad
  - Curvas ROC y área bajo la curva (AUC-ROC)
  - Matriz de confusión
  - Validación cruzada
- Preprocesamiento de datos:
  - Normalización y estandarización
  - Manejo de datos faltantes
  - Ingeniería de características
  - Selección de características
- Optimización de modelos:
  - Hiperparámetros y búsqueda en cuadrícula
  - Optimización bayesiana
  - Regularización
  - Redes neuronales convolucionales (CNN) y recurrentes (RNN)
- Aprendizaje por refuerzo:
  - Q-Learning
  - Algoritmos de políticas
  - Exploración y explotación

    - Funciones de valor

- Ética en el machine learning:
    - Sesgo y equidad
    - Transparencia y explicabilidad
    - Privacidad y seguridad
    - Responsabilidad en el despliegue de modelos

11.1. SVM

1. **Hiperplano:** En un espacio de características n-dimensional, un hiperplano es un subespacio de dimensión $n-1$. Para un problema de clasificación binaria, un hiperplano divide el espacio en dos regiones, asignando puntos a una clase u otra.

2. **Margen:** El margen es la distancia perpendicular desde el hiperplano a los puntos más cercanos de cada clase. SVM busca el hiperplano que maximiza este margen, lo que se traduce en una mayor robustez y generalización del modelo.

3. **Vectores de Soporte:** Estos son los puntos de datos más cercanos al hiperplano y tienen un papel crucial en la definición del margen. Cambiar estos vectores de soporte afecta directamente al modelo, y son los únicos puntos que importan para la determinación del hiperplano.

4. **Función de Decisión:** La función de decisión de SVM es el hiperplano que se utiliza para clasificar nuevos puntos de datos. Dada una entrada, la función de decisión evalúa de qué lado del hiperplano cae el punto y asigna la etiqueta correspondiente.

5. **Kernel Trick:** SVM puede manejar eficientemente datos no lineales mediante el uso de funciones de kernel. Estas funciones transforman el espacio de características original en uno de mayor dimensión, permitiendo así que los datos sean separados de manera no lineal en el espacio transformado.

6. **Parámetros de SVM:**
    - **C (Parámetro de Regularización):** Controla el equilibrio entre tener un margen más amplio y clasificar correctamente los puntos de entrenamiento.
    - **Kernel:** Define la función de kernel utilizada (lineal, polinómica, radial, etc.).
    - **Gamma (para kernels no lineales):** Controla el alcance de influencia de un solo punto de datos en la decisión.

7. **Proceso de Entrenamiento:** Dado un conjunto de datos de entrenamiento etiquetado, SVM busca el hiperplano óptimo que maximiza el margen entre las clases. Esto se realiza a través de técnicas de optimización cuadrática.

8. **SVM para Regresión:** Además de la clasificación, SVM se puede utilizar para problemas de regresión. En este caso, el objetivo es ajustar un hiperplano de modo que contenga la mayor cantidad posible de puntos dentro de un margen predefinido.

9. **Ventajas y Desventajas:**

   - **Ventajas:** Efectivo en espacios de alta dimensión, eficaz en conjuntos de datos pequeños y versátil gracias al kernel trick.
   - **Desventajas:** Sensible a la escala de las características, puede ser computacionalmente costoso para grandes conjuntos de datos y requiere la elección cuidadosa de parámetros.

10. **Aplicaciones:** SVM se utiliza en una variedad de campos, como reconocimiento de escritura, clasificación de imágenes, diagnóstico médico, entre otros.

11.1.1. Elementos Matemáticos de las SVM

1. **Hiperplano:** Un hiperplano se define como $w \cdot x - b = 0$, donde w es el vector de pesos, x es el vector de entrada, y b es el sesgo.

2. **Margen:** El margen M entre un hiperplano y un punto x_i se define como $M = \frac{1}{\|w\|}|w \cdot x_i - b|$.

3. **Vectores de Soporte:** Los vectores de soporte son los puntos x_i que cumplen la condición $|w \cdot x_i - b| = 1/\|w\|$.

4. **Función de Decisión:** La función de decisión es $f(x) = w \cdot x - b$. Si $f(x) > 0$, el punto x se clasifica como clase 1; si $f(x) < 0$, se clasifica como clase -1.

5. **Kernel Trick:** La función de kernel $K(x, x')$ representa el producto escalar en un espacio de características de mayor dimensión. Ejemplos comunes incluyen el kernel lineal ($K(x, x') = x \cdot x'$), kernel polinómico ($K(x, x') = (x \cdot x' + 1)^d$), y kernel radial ($K(x, x') = \exp(-\gamma\|x - x'\|^2)$).

6. **Parámetros de SVM:**

   - C (Parámetro de Regularización): Se introduce en la función de pérdida para controlar el equilibrio entre tener un margen más amplio y clasificar correctamente los puntos de entrenamiento.
   - w (Vector de Pesos): Aprende durante el entrenamiento y define la orientación del hiperplano.
   - b (Sesgo): Parámetro de ajuste del hiperplano.
   - γ (para kernels no lineales): Controla el alcance de influencia de un solo punto de datos en la decisión.

7. **Proceso de Entrenamiento:** El proceso de entrenamiento implica la minimización de la función de pérdida, que incluye el término de regularización $C\|w\|^2$ y la función de pérdida hinge.

8. **SVM para Regresión:** Para regresión, el objetivo es ajustar un hiperplano de modo que $|w \cdot x_i - b| \leq \epsilon$ para puntos de entrenamiento x_i.

9. **Ventajas y Desventajas:**

   - Ventajas: Efectivo en espacios de alta dimensión, eficaz en conjuntos de datos pequeños y versátil gracias al kernel trick.
   - Desventajas: Sensible a la escala de las características, puede ser computacionalmente costoso para grandes conjuntos de datos y requiere la elección cuidadosa de parámetros.

10. **Aplicaciones:** SVM se aplica en una variedad de problemas, como reconocimiento de escritura, clasificación de imágenes y diagnóstico médico.

11.2. Árboles de Decisión

1. **Definición:** Un árbol de decisión es una estructura jerárquica en forma de árbol que se utiliza para representar decisiones y sus posibles consecuencias. Cada nodo interno del árbol representa una prueba en una característica, cada rama representa un resultado posible de la prueba, y cada hoja representa un resultado final o una decisión.

2. **Proceso de Construcción:** El árbol se construye de manera recursiva. En cada paso, se elige la mejor característica para dividir el conjunto de datos en función de algún criterio, como la ganancia de información o la impureza de Gini. Este proceso se repite hasta que se alcanza algún criterio de parada, como la profundidad máxima del árbol o un número mínimo de puntos en una hoja.

3. **Criterios de División:** Los criterios comunes para la división incluyen:

   - **Ganancia de Información:** Mide cuánta información nueva proporciona una característica.
   - **Impureza de Gini:** Mide la probabilidad de clasificar incorrectamente un elemento si es etiquetado aleatoriamente.

4. **Ventajas y Desventajas:**

   - **Ventajas:** Fácil interpretación, no requiere normalización de datos, manejo natural de características categóricas.
   - **Desventajas:** Propenso al sobreajuste, especialmente en conjuntos de datos pequeños y ruidosos.

11.2.1. Bosques Aleatorios

1. **Definición:** Un bosque aleatorio es una colección de árboles de decisión entrenados en subconjuntos aleatorios del conjunto de datos y utilizando técnicas de agregación para mejorar la precisión y controlar el sobreajuste.

2. **Proceso de Construcción:** Se crean múltiples árboles de decisión utilizando diferentes subconjuntos del conjunto de datos de entrenamiento y características aleatorias en cada división. Luego, las predicciones de cada árbol se promedian (en regresión) o se votan (en clasificación) para obtener la predicción final del bosque.

3. **Técnica de Bagging:** La construcción de árboles en subconjuntos aleatorios del conjunto de datos se conoce como bagging (bootstrap aggregating). Esto ayuda a reducir la varianza y evitar el sobreajuste al promediar los errores.

4. **Importancia de Características:** Los bosques aleatorios proporcionan una medida de la importancia de las características, que indica cuánto contribuye cada característica a la precisión del modelo. Esto es útil para la selección de características.

5. **Ventajas y Desventajas:**

   - **Ventajas:** Reducción del sobreajuste en comparación con un solo árbol, manejo automático del sobreajuste, buen rendimiento en conjuntos de datos grandes y complejos.
   - **Desventajas:** Menos interpretables que los árboles de decisión individuales.

11.2.2. Elementos Matemáticos de Árboles de Decisión

1. **Definición:** Un árbol de decisión se representa como una función $T(x)$ que asigna una instancia x a una hoja del árbol. Cada nodo interno j realiza una prueba en una característica $f_j(x)$, y cada rama representa una condición de prueba. La estructura del árbol se define por las funciones indicadoras $I(x, j)$ que indican si la instancia x llega al nodo j.

$$T(x) = \sum_{j=1}^{J} I(x, j) \cdot C_j$$

   Donde J es el número de nodos, C_j es el valor en la hoja correspondiente al nodo j, y $I(x, j)$ es 1 si la instancia x llega al nodo j y 0 de lo contrario.

2. **Proceso de Construcción:** La construcción del árbol implica seleccionar la mejor característica f_j y el umbral t_j en cada nodo j para maximizar la ganancia de información o reducir la impureza de Gini.

$$\text{Ganancia de Información: } Gain(D, j, t) = H(D) - \frac{N_L}{N} H(D_L) - \frac{N_R}{N} H(D_R)$$

$$\text{Impureza de Gini: } Gini(D) = 1 - \sum_{k=1}^{K} \left(\frac{|C_k|}{|D|} \right)^2$$

   Donde D es el conjunto de datos en el nodo, D_L y D_R son los conjuntos de datos en los nodos izquierdo y derecho después de la división, N es el número total de instancias en D, y N_L y N_R son los números de instancias en los nodos izquierdo y derecho.

3. **Ventajas y Desventajas:**
   - **Ventajas:** Fácil interpretación, no requiere normalización de datos, manejo natural de características categóricas.
   - **Desventajas:** Propenso al sobreajuste, especialmente en conjuntos de datos pequeños y ruidosos.

11.2.3. Bosques Aleatorios

1. **Definición:** Un bosque aleatorio es una colección de B árboles de decisión $T_b(x)$, donde cada árbol se entrena en un subconjunto aleatorio de los datos de entrenamiento. La predicción se obtiene promediando (en regresión) o votando (en clasificación) las predicciones individuales de los árboles.

$$\text{Predicción del Bosque: } \hat{Y}(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$$

2. **Proceso de Construcción:** Cada árbol en el bosque se construye utilizando bagging, que consiste en seleccionar aleatoriamente un subconjunto de las instancias de entrenamiento con reemplazo. Además, en cada división de nodo, se selecciona un subconjunto aleatorio de características.

$$\text{Bagging: } D_b = \{(x_i, y_i)\} \text{ con } i \sim \text{Uniforme}(1, N)$$

$$\text{Características Aleatorias: } f_j \text{ con } j \sim \text{Uniforme}(1, P)$$

3. **Importancia de Características:** La importancia de la característica f_j se mide mediante la disminución promedio en la ganancia de impureza o la reducción en el error cuadrático medio cuando se utiliza f_j para dividir los nodos a lo largo de todos los árboles.

$$\text{Importancia de } f_j = \frac{1}{B}\sum_{b=1}^{B} \sum_{j \text{ en } T_b} \text{Importancia de } f_j \text{ en } T_b$$

12. Referencias

- Libros:
  1. "The Hundred-Page Machine Learning Book"por Andriy Burkov
  2. "Machine Learning: A Probabilistic Perspective"por Kevin P. Murphy
  3. "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow"por Aurélien Géron

4. "Pattern Recognition and Machine Learning"por Christopher M. Bishop
5. "Deep Learning"por Ian Goodfellow, Yoshua Bengio y Aaron Courville

- Cursos:

1. "Machine Learning"por Andrew Ng en Coursera
2. .ªpplied Data Science with Python"por la Universidad de Míchigan en Coursera
3. "Deep Learning Specialization"por Andrew Ng en Coursera
4. "Machine Learning Crash Course"por Google
5. Ïntroduction to Machine Learning"por la Universidad de Columbia en edX

- Sitios web:

1. Kaggle
2. GitHub
3. Towards Data Science
4. Machine Learning Mastery
5. Google AI

Fuentes:

1. Machine Learning citation style [Update October 2023] - Paperpile
2. ICML2021 Template - Overleaf, Online LaTeX Editor
3. Machine Learning — Submission guidelines - Springer
4. Journal of Machine Learning Research

13. Bioinformatics: New revision

13.1. A bayesian framework for combining gene predictions - Pavlovic

Biology and biotechnology are undergoing a techonolgical revolution which is transforming research into an information-rich enterprise. A typical bacterial genome sequence is comprised of several million bases of DNA and contains several thousand genes. The human genome is approximately 3 billion bases long and it contains approximately 30,000 putative genes identified thus far.

14. Random Forest: Explanation in Mathematical Terms

Random Forest is a supervised learning algorithm used for both classification and regression problems. The main idea behind Random Forest is to build multiple decision trees during training and combine their results to obtain a more robust and accurate prediction.

Let's assume we have a training dataset $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, where x_i represents the features and y_i is the target variable.

1. **Construction of Decision Trees:**

   - For each tree, a random subset of features is selected (randomly sample features), and a random subset of training data is chosen (randomly sample observations with replacement).
   - A decision tree is constructed using the subset of data and features.
   - The decision tree is represented as $h_i(x; \theta_i)$, where i denotes the tree index, x is the input feature vector, and θ_i represents the parameters of the tree.
   - At each node t of the tree, a feature j_t is selected from the random subset, and the split is determined based on minimizing impurity:

     $$\theta_{i,t} = \arg\min_{j_t, s_t} [\text{Impurity}(D_t) - p_{\text{left}}\text{Impurity}(D_{\text{left}}) - p_{\text{right}}\text{Impurity}(D_{\text{right}})]$$

     where D_t is the dataset at node t, D_{left} and D_{right} are the datasets in the left and right child nodes, p_{left} and p_{right} are the proportions of data in the left and right child nodes, and $\text{Impurity}(\cdot)$ is a measure of impurity, such as Gini index for classification or mean squared error for regression.

2. **Voting or Averaging:**

   - For classification problems, the final prediction is obtained by majority voting among the trees:

     $$H(x) = \text{mode}\{h_1(x; \theta_1), h_2(x; \theta_2), \ldots, h_n(x; \theta_n)\}$$

   - For regression problems, the final prediction is the average of predictions from all trees:

     $$H(x) = \frac{1}{n} \sum_{i=1}^{n} h_i(x; \theta_i)$$

3. **Variance Reduction:**

   - By building decision trees in a random manner, the variance of the model is reduced, leading to a more robust and generalizable model.

The final prediction is obtained by combining the predictions of all trees.

15. Deep Learning: Overview

Deep Learning is a subfield of machine learning that focuses on neural networks with multiple layers (deep neural networks). These networks can automatically learn hierarchical representations of data, allowing them to capture intricate patterns and features.

1. **Neural Network Representation:**
   - A neural network consists of layers of interconnected nodes (neurons) organized into an input layer, one or more hidden layers, and an output layer.
   - The input layer represents the features of the data, and each neuron in the layer processes a specific feature.
   - Hidden layers perform nonlinear transformations on the input data, learning hierarchical representations.
   - The output layer produces the final prediction based on the learned representations.
2. **Training a Neural Network:**
   - During training, the network's parameters (weights and biases) are adjusted to minimize the difference between predicted and actual outputs.
   - This optimization is typically done using backpropagation and gradient descent.
3. **Activation Functions:**
   - Activation functions introduce nonlinearity to the neural network, enabling it to learn complex patterns.
   - Common activation functions include ReLU (Rectified Linear Unit), Sigmoid, and Hyperbolic Tangent (tanh).
4. **Forward Pass:**
   - The forward pass of a neural network involves computing the output of the network for a given input.
   - Given an input vector x, the output y is computed by passing x through the network's layers using learned weights and biases.
   - The output of each layer is computed as:

     $$a^{(l)} = W^{(l)} a^{(l-1)} + b^{(l)}$$

     where $W^{(l)}$ is the weight matrix, $a^{(l)}$ is the activation of layer l, $a^{(l-1)}$ is the activation of the previous layer, and $b^{(l)}$ is the bias vector.
   - The activation function is then applied to $a^{(l)}$ to introduce nonlinearity.
5. **Loss Function:**

- The loss function measures the difference between the predicted output and the true output.
- Common loss functions include mean squared error for regression and cross-entropy for classification.
- The goal during training is to minimize the loss by adjusting the network's parameters.

6. **Backpropagation:**

- Backpropagation is a training algorithm that computes the gradient of the loss with respect to the network's parameters.
- The gradient is used to update the parameters in the direction that reduces the loss.
- It involves computing the gradient of the loss with respect to the output of each layer and using the chain rule to propagate these gradients backward through the network.

16. Example of Implementation in R:

16.1. Install and Load the Keras Library:

```
install.packages("keras")
library(keras)
```

16.2. Build a Simple Neural Network:

```
model <- keras_model_sequential() %>%
  layer_dense(units = 128, activation = 'relu', input_shape = c(10)) %>%
  layer_dense(units = 1, activation = 'sigmoid')

summary(model)
```

16.3. Compile and Train the Neural Network:

```
model %>% compile(
  optimizer = 'adam',
  loss = 'binary_crossentropy',
  metrics = c('accuracy')
)

# Assuming you have training data X_train and labels y_train
history <- model %>% fit(
  X_train, y_train,
  epochs = 10, batch_size = 32,
```

```
  validation_split = 0.2
)
```

16.4. Evaluate the Model:

```
# Assuming you have test data X_test and labels y_test
evaluate_result <- model %>% evaluate(X_test, y_test)
print(evaluate_result)
```

17. Impurity Measures in Random Forest

In the context of Random Forest, impurity measures play a crucial role in the construction of decision trees. The two commonly used impurity measures are the Gini index for classification and the mean squared error for regression.

17.1. Gini Index for Classification

The Gini index is a measure of impurity used in classification problems. Given a node in a decision tree that contains data points from different classes, the Gini index quantifies how often a randomly chosen data point would be incorrectly classified.

For a node t with K classes and a set of data points D_t, the Gini index ($Gini(t)$) is calculated as follows:

$$Gini(t) = 1 - \sum_{i=1}^{K} p_i^2$$

where p_i is the proportion of data points in class i at node t. A lower Gini index indicates a purer node with predominantly one class.

In the context of Random Forest, the decision tree split is determined by minimizing the weighted sum of Gini indices for the left and right child nodes. The split that results in the lowest overall Gini index is chosen.

17.2. Mean Squared Error for Regression

For regression problems, the impurity measure used is the mean squared error (MSE). Unlike classification, where impurity is related to the purity of classes in a node, regression impurity is a measure of the variability of target values within a node.

For a node t with data points D_t, the MSE ($MSE(t)$) is calculated as follows:

$$MSE(t) = \frac{1}{|D_t|} \sum_{i \in D_t} (y_i - \bar{y}_t)^2$$

where y_i is the target value of data point i, $|D_t|$ is the number of data points in node t, and $\bar{y}_t$ is the mean target value of all data points in node t.

Similar to the Gini index, in Random Forest, the decision tree split is determined by minimizing the weighted sum of MSE for the left and right child nodes.

These impurity measures guide the construction of individual decision trees within the Random Forest ensemble, contributing to the overall robustness and predictive power of the model.

18. Construcción del Árbol de Decisión

18.1. Teoría:

Un árbol de decisión es una estructura de datos que representa un conjunto de decisiones y sus posibles consecuencias. En el contexto de Random Forest, la construcción de un árbol de decisión sigue el principio de .ªprendizaje supervisado", donde el algoritmo aprende patrones a partir de un conjunto de datos etiquetado.

La construcción del árbol se realiza a través de divisiones recursivas basadas en características del conjunto de datos. Cada nodo del árbol representa una pregunta sobre una característica, y las ramas que surgen de ese nodo son las respuestas a esa pregunta. El proceso continúa hasta que se alcanza un criterio de parada, como la profundidad máxima del árbol o el número mínimo de muestras en un nodo.

18.2. Elementos Matemáticos:

1. **Función de Impureza:**

En cada nodo del árbol, se elige la característica y el umbral que minimizan la impureza en los nodos hijos resultantes. La impureza se mide mediante funciones como el Índice de Gini para clasificación o el Error Cuadrático Medio (MSE) para regresión.

Para clasificación:

$$Gini(t) = 1 - \sum_{i=1}^{K} p_i^2$$

Donde p_i es la proporción de ejemplos de la clase i en el nodo t.

Para regresión:

$$MSE(t) = \frac{1}{|D_t|} \sum_{i \in D_t} (y_i - \bar{y}_t)^2$$

Donde D_t es el conjunto de datos en el nodo t, y_i es la etiqueta del ejemplo i, y $\bar{y}_t$ es la media de las etiquetas en el nodo t.

2. **Criterio de División:**

La elección de la mejor característica y umbral se basa en la reducción de la impureza. Se busca el par (j, s) que minimiza la expresión:

$$\theta_{i,t} = \arg \min_{j,s} [\text{Impureza}(D_t) - p_{\text{left}} \text{Impureza}(D_{\text{left}}) - p_{\text{right}} \text{Impureza}(D_{\text{right}})]$$

Donde D_t es el conjunto de datos en el nodo t, D_{left} y D_{right} son los conjuntos de datos en los nodos izquierdo y derecho después de la división, y p_{left} y p_{right} son las proporciones de datos en esos nodos.

3. **Criterios de Parada:**

Para evitar sobreajuste, se utilizan criterios de parada, como la profundidad máxima del árbol o el número mínimo de muestras requeridas para realizar una división.

La construcción de cada árbol en Random Forest implica este proceso iterativo y se repite para cada árbol en el bosque. La diversidad en la construcción de árboles se logra mediante el uso de diferentes subconjuntos aleatorios de características y datos en cada árbol. La combinación de predicciones de estos árboles mejora la generalización del modelo y su capacidad predictiva en nuevos datos.

19. Reducción de Varianza en Random Forest

19.1. Teoría:

La reducción de la varianza es uno de los objetivos clave en la construcción de árboles de decisión en Random Forest. Esta técnica busca mejorar la generalización del modelo al reducir la sensibilidad a pequeñas variaciones en los datos de entrenamiento.

La varianza se refiere a la variabilidad de las predicciones de un modelo respecto a diferentes conjuntos de datos de entrenamiento. En Random Forest, la reducción de la varianza se logra mediante dos técnicas principales: Bagging y Random Subspace.

19.1.1. Bagging (Bootstrap Aggregating):

Bagging es una técnica que consiste en entrenar múltiples modelos en diferentes subconjuntos de datos generados mediante muestreo con reemplazo (bootstrap). Cada árbol de decisión en Random Forest se entrena en un conjunto de datos ligeramente diferente, lo que introduce diversidad en los modelos.

19.1.2. Random Subspace:

Random Subspace es otra técnica utilizada para reducir la correlación entre los árboles de decisión. En lugar de usar todas las características para cada árbol, se selecciona un subconjunto aleatorio de características para entrenar cada árbol. Esto ayuda a que cada árbol se especialice en diferentes aspectos de los datos, mejorando la diversidad y reduciendo la correlación entre las predicciones.

19.2. Elementos Matemáticos:

La reducción de la varianza no se expresa directamente con fórmulas matemáticas específicas, pero los conceptos clave son fundamentales:

1. **Bagging:** - Cada árbol T_i se entrena en un conjunto de datos D_i generado por muestreo con reemplazo (bootstrap). - La predicción final se obtiene promediando las predicciones de todos los árboles:

$$H(x) = \frac{1}{N} \sum_{i=1}^{N} T_i(x)$$

2. **Random Subspace:** - Cada árbol T_i se entrena utilizando un subconjunto aleatorio de características. - La predicción final se obtiene promediando las predicciones de todos los árboles:

$$H(x) = \frac{1}{N} \sum_{i=1}^{N} T_i(x)$$

Estos enfoques combinados contribuyen a reducir la varianza del modelo, lo que resulta en un modelo más robusto y generalizable.

20. Votación y Promedio en Random Forest

20.1. Teoría:

La técnica de "Votación"(para problemas de clasificación) o "Promedio"(para problemas de regresión) es crucial en Random Forest para combinar las predicciones de múltiples árboles de decisión y obtener una predicción final más robusta y precisa.

20.1.1. Para Problemas de Clasificación:

En problemas de clasificación, el enfoque de votación se utiliza. Cada árbol en el bosque emite una predicción de clase, y la clase final se determina por mayoría de votos.

20.1.2. Para Problemas de Regresión:

En problemas de regresión, se utiliza un enfoque de promedio. Cada árbol realiza una predicción numérica, y la predicción final es el promedio de todas las predicciones.

20.2. Elementos Matemáticos:

1. **Votación para Clasificación:** - La predicción final para un ejemplo x se obtiene por mayoría de votos:

$$H(x) = \text{mode}\{T_1(x), T_2(x), ..., T_N(x)\}$$

2. **Promedio para Regresión:** - La predicción final para un ejemplo x se obtiene promediando las predicciones de todos los árboles:

$$H(x) = \frac{1}{N} \sum_{i=1}^{N} T_i(x)$$

Donde $T_i(x)$ representa la predicción del árbol i para el ejemplo x, y N es el número total de árboles en el bosque.

Estos enfoques de votación y promedio permiten que Random Forest combine la información de múltiples árboles de decisión, mejorando la generalización y la capacidad predictiva del modelo.

21. Resumen

Random Forest es un algoritmo de aprendizaje automático que se utiliza para resolver problemas de clasificación y regresión. En lugar de utilizar un único árbol de decisión, Random Forest construye varios árboles de decisión y los combina para obtener una predicción más precisa. Cada árbol de decisión se construye utilizando un subconjunto aleatorio de las características del conjunto de datos original. Al final, las predicciones de cada árbol se promedian para obtener una predicción final.

El algoritmo de Random Forest se basa en dos técnicas: Bagging y Random Subspace. Bagging es una técnica que se utiliza para reducir la varianza de un modelo al entrenar múltiples modelos en diferentes subconjuntos de datos. Random Subspace es una técnica que se utiliza para reducir la correlación entre los modelos al entrenar cada modelo en diferentes subconjuntos de características.

Aquí hay algunos elementos matemáticos que se utilizan en Random Forest:

- **Árbol de decisión:** Un árbol de decisión es una estructura de datos que se utiliza para modelar decisiones y sus posibles consecuencias. Cada nodo en el árbol representa una decisión, y cada rama representa una posible consecuencia de esa decisión. Los árboles de decisión se construyen utilizando un conjunto de reglas que se utilizan para tomar decisiones.
- **Bootstrap:** Bootstrap es una técnica que se utiliza para generar múltiples conjuntos de datos a partir de un conjunto de datos original. Cada conjunto de datos se genera mediante muestreo con reemplazo, lo que significa que cada elemento del conjunto de datos original tiene la misma probabilidad de ser seleccionado en cada conjunto de datos generado.
- **Out-of-Bag Error:** Out-of-Bag Error es una técnica que se utiliza para estimar el error de validación de un modelo sin la necesidad de un conjunto de datos de validación separado. El error se estima utilizando los datos que no se incluyeron en el conjunto de datos de entrenamiento para cada árbol de decisión.

21.1. Ejemplo en R: Random Forest

Aquí hay un ejemplo de cómo construir un modelo de bosque aleatorio en R:

```
# Cargamos el paquete necesario para este ejemplo
library(randomForest)

# Cargamos el conjunto de datos que deseamos utilizar
data(iris)

# Dividimos el conjunto de datos en conjuntos de entrenamiento y prueba
trainIndex <- createDataPartition(iris$Species, p = .8, list = FALSE, times = 1)

# Entrenamos el modelo de bosque aleatorio
```

```
rf_model <- randomForest(Species ~ ., data = iris[trainIndex,])

# Realizamos predicciones en el conjunto de prueba
predictions <- predict(rf_model, iris[-trainIndex,])
```

21.2. Deep Learning

Deep Learning es un subcampo del aprendizaje automático que se centra en la creación de redes neuronales artificiales profundas. Estas redes neuronales están diseñadas para imitar el cerebro humano y son capaces de aprender patrones complejos en los datos.

Aquí hay un ejemplo de cómo construir una red neuronal profunda en R utilizando el paquete keras:

```
# Cargamos los paquetes necesarios
library(keras)
library(tensorflow)

# Cargamos el conjunto de datos que deseamos utilizar
data(iris)

# Dividimos el conjunto de datos en conjuntos de entrenamiento y prueba
trainIndex <- createDataPartition(iris$Species, p = .8, list = FALSE, times = 1)

# Creamos la red neuronal
model <- keras_model_sequential() %>%
  layer_dense(units = 4, input_shape = c(4)) %>%
  layer_activation("relu") %>%
  layer_dense(units = 3) %>%
  layer_activation("softmax")

# Compilamos la red neuronal
model %>% compile(loss = "categorical_crossentropy", optimizer = "adam", metrics = "accu

# Entrenamos la red neuronal
history <- model %>% fit(
  x = iris[trainIndex, 1:4],
  y = to_categorical(as.numeric(iris[trainIndex, 5])),
  epochs = 100,
  batch_size = 10,
  validation_split = 0.2
)
```

22. Algoritmos de Aprendizaje Supervisado

22.1. Regresión Lineal y Logística

22.1.1. Regresión Lineal

La regresión lineal es un modelo que busca modelar la relación lineal entre una variable dependiente Y y una o más variables independientes X. El modelo se define como:

$$Y = \beta_0 + \beta_1 X + \varepsilon$$

donde:

- Y es la variable dependiente que queremos predecir.
- X es la variable independiente que utilizamos para la predicción.
- β_0 es la ordenada al origen, que representa el valor de Y cuando X es cero.
- β_1 es la pendiente de la recta, que indica cuánto cambia Y por un cambio unitario en X.
- ε es el término de error que captura la variabilidad no explicada por el modelo.

El objetivo es encontrar los valores de β_0 y β_1 que minimizan la suma de los cuadrados de los errores ε:

$$\min_{\beta_0,\beta_1} \sum_{i=1}^{n} (Y_i - (\beta_0 + \beta_1 X_i))^2$$

Esto se puede hacer utilizando técnicas como el método de mínimos cuadrados.

22.1.2. Regresión Logística

La regresión logística es un modelo utilizado para problemas de clasificación binaria. Se emplea la función logística, también conocida como la función sigmoide, para transformar la salida de la regresión lineal en un valor entre 0 y 1, que se interpreta como la probabilidad de pertenecer a la clase positiva. La función sigmoide está definida como:

$$P(Y = 1) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$

donde:

- $P(Y = 1)$ es la probabilidad de pertenecer a la clase positiva.
- e es la base del logaritmo natural.

El modelo de regresión logística busca encontrar los valores de β_0 y β_1 que maximizan la función de verosimilitud. La función de verosimilitud es el producto de las probabilidades condicionales de observar las etiquetas de clase dados los valores de X:

$$\max_{\beta_0,\beta_1} \mathcal{L}(\beta_0, \beta_1) = \prod_{i=1}^{n} P(Y_i)^{y_i} \cdot (1 - P(Y_i))^{1-y_i}$$

donde:

- $\mathcal{L}(\beta_0, \beta_1)$ es la función de verosimilitud.
- y_i es la etiqueta de la clase para la observación i.

La regresión logística se ajusta típicamente maximizando la log-verosimilitud para obtener estimaciones de β_0 y β_1.

22.1.3. Implementacion

```
# Ejemplo de Regresión Lineal
set.seed(123)
# Crear datos de ejemplo
X <- rnorm(100)
Y <- 2 * X + rnorm(100)

# Ajustar el modelo de regresión lineal
modelo_lineal <- lm(Y ~ X)

# Imprimir resultados
summary(modelo_lineal)

# Graficar el modelo
plot(X, Y, main = "Regresión Lineal", xlab = "X", ylab = "Y")
abline(modelo_lineal, col = "red")

# Ejemplo de Regresión Logística
set.seed(123)
# Crear datos de ejemplo para clasificación binaria
X <- rnorm(100)
probabilidades <- exp(2 * X) / (1 + exp(2 * X))
Y_binario <- rbinom(100, 1, probabilidades)

# Ajustar el modelo de regresión logística
modelo_logistico <- glm(Y_binario ~ X, family = binomial)

# Imprimir resultados
summary(modelo_logistico)
```

```
# Graficar el modelo
plot(X, Y_binario, main = "Regresión Logística",
         xlab = "X", ylab = "Y", col = Y_binario + 1)
curve(predict(modelo_logistico,
           data.frame(X = x), type = "response"),
           add = TRUE, col = "red")
```

22.2. Máquinas de Soporte Vectorial (SVM)

1. **Hiperplano:** En un espacio de características n-dimensional, un hiperplano es un subespacio de dimensión $n-1$. Para un problema de clasificación binaria, un hiperplano divide el espacio en dos regiones, asignando puntos a una clase u otra.

2. **Margen:** El margen es la distancia perpendicular desde el hiperplano a los puntos más cercanos de cada clase. SVM busca el hiperplano que maximiza este margen, lo que se traduce en una mayor robustez y generalización del modelo.

3. **Vectores de Soporte:** Estos son los puntos de datos más cercanos al hiperplano y tienen un papel crucial en la definición del margen. Cambiar estos vectores de soporte afecta directamente al modelo, y son los únicos puntos que importan para la determinación del hiperplano.

4. **Función de Decisión:** La función de decisión de SVM es el hiperplano que se utiliza para clasificar nuevos puntos de datos. Dada una entrada, la función de decisión evalúa de qué lado del hiperplano cae el punto y asigna la etiqueta correspondiente.

5. **Kernel Trick:** SVM puede manejar eficientemente datos no lineales mediante el uso de funciones de kernel. Estas funciones transforman el espacio de características original en uno de mayor dimensión, permitiendo así que los datos sean separados de manera no lineal en el espacio transformado.

6. **Parámetros de SVM:**

   - **C (Parámetro de Regularización):** Controla el equilibrio entre tener un margen más amplio y clasificar correctamente los puntos de entrenamiento.
   - **Kernel:** Define la función de kernel utilizada (lineal, polinómica, radial, etc.).
   - **Gamma (para kernels no lineales):** Controla el alcance de influencia de un solo punto de datos en la decisión.

7. **Proceso de Entrenamiento:** Dado un conjunto de datos de entrenamiento etiquetado, SVM busca el hiperplano óptimo que maximiza el margen entre las clases. Esto se realiza a través de técnicas de optimización cuadrática.

8. **SVM para Regresión:** Además de la clasificación, SVM se puede utilizar para problemas de regresión. En este caso, el objetivo es ajustar un hiperplano de modo que contenga la mayor cantidad posible de puntos dentro de un margen predefinido.

9. **Ventajas y Desventajas:**
    - **Ventajas:** Efectivo en espacios de alta dimensión, eficaz en conjuntos de datos pequeños y versátil gracias al kernel trick.
    - **Desventajas:** Sensible a la escala de las características, puede ser computacionalmente costoso para grandes conjuntos de datos y requiere la elección cuidadosa de parámetros.
10. **Aplicaciones:** SVM se utiliza en una variedad de campos, como reconocimiento de escritura, clasificación de imágenes, diagnóstico médico, entre otros.

22.2.1. Elementos Matemáticos de las SVM

1. **Hiperplano:** Un hiperplano se define como $w \cdot x - b = 0$, donde w es el vector de pesos, x es el vector de entrada, y b es el sesgo.
2. **Margen:** El margen M entre un hiperplano y un punto x_i se define como $M = \frac{1}{\|w\|}|w \cdot x_i - b|$.
3. **Vectores de Soporte:** Los vectores de soporte son los puntos x_i que cumplen la condición $|w \cdot x_i - b| = 1/\|w\|$.
4. **Función de Decisión:** La función de decisión es $f(x) = w \cdot x - b$. Si $f(x) > 0$, el punto x se clasifica como clase 1; si $f(x) < 0$, se clasifica como clase -1.
5. **Kernel Trick:** La función de kernel $K(x, x')$ representa el producto escalar en un espacio de características de mayor dimensión. Ejemplos comunes incluyen el kernel lineal ($K(x, x') = x \cdot x'$), kernel polinómico ($K(x, x') = (x \cdot x' + 1)^d$), y kernel radial ($K(x, x') = \exp(-\gamma\|x - x'\|^2)$).
6. **Parámetros de SVM:**
    - C (Parámetro de Regularización): Se introduce en la función de pérdida para controlar el equilibrio entre tener un margen más amplio y clasificar correctamente los puntos de entrenamiento.
    - w (Vector de Pesos): Aprende durante el entrenamiento y define la orientación del hiperplano.
    - b (Sesgo): Parámetro de ajuste del hiperplano.
    - γ (para kernels no lineales): Controla el alcance de influencia de un solo punto de datos en la decisión.
7. **Proceso de Entrenamiento:** El proceso de entrenamiento implica la minimización de la función de pérdida, que incluye el término de regularización $C\|w\|^2$ y la función de pérdida hinge.
8. **SVM para Regresión:** Para regresión, el objetivo es ajustar un hiperplano de modo que $|w \cdot x_i - b| \leq \epsilon$ para puntos de entrenamiento x_i.

9. **Ventajas y Desventajas:**
    - Ventajas: Efectivo en espacios de alta dimensión, eficaz en conjuntos de datos pequeños y versátil gracias al kernel trick.
    - Desventajas: Sensible a la escala de las características, puede ser computacionalmente costoso para grandes conjuntos de datos y requiere la elección cuidadosa de parámetros.
10. **Aplicaciones:** SVM se aplica en una variedad de problemas, como reconocimiento de escritura, clasificación de imágenes y diagnóstico médico.

22.3. Árboles de Decisión y Bosques Aleatorios

1. **Definición:** Un árbol de decisión es una estructura jerárquica en forma de árbol que se utiliza para representar decisiones y sus posibles consecuencias. Cada nodo interno del árbol representa una prueba en una característica, cada rama representa un resultado posible de la prueba, y cada hoja representa un resultado final o una decisión.
2. **Proceso de Construcción:** El árbol se construye de manera recursiva. En cada paso, se elige la mejor característica para dividir el conjunto de datos en función de algún criterio, como la ganancia de información o la impureza de Gini. Este proceso se repite hasta que se alcanza algún criterio de parada, como la profundidad máxima del árbol o un número mínimo de puntos en una hoja.
3. **Criterios de División:** Los criterios comunes para la división incluyen:
    - **Ganancia de Información:** Mide cuánta información nueva proporciona una característica.
    - **Impureza de Gini:** Mide la probabilidad de clasificar incorrectamente un elemento si es etiquetado aleatoriamente.
4. **Ventajas y Desventajas:**
    - **Ventajas:** Fácil interpretación, no requiere normalización de datos, manejo natural de características categóricas.
    - **Desventajas:** Propenso al sobreajuste, especialmente en conjuntos de datos pequeños y ruidosos.

22.3.1. Bosques Aleatorios

1. **Definición:** Un bosque aleatorio es una colección de árboles de decisión entrenados en subconjuntos aleatorios del conjunto de datos y utilizando técnicas de agregación para mejorar la precisión y controlar el sobreajuste.
2. **Proceso de Construcción:** Se crean múltiples árboles de decisión utilizando diferentes subconjuntos del conjunto de datos de entrenamiento y características aleatorias en cada división. Luego, las predicciones de cada árbol se promedian (en regresión) o se votan (en clasificación) para obtener la predicción final del bosque.

3. **Técnica de Bagging:** La construcción de árboles en subconjuntos aleatorios del conjunto de datos se conoce como bagging (bootstrap aggregating). Esto ayuda a reducir la varianza y evitar el sobreajuste al promediar los errores.

4. **Importancia de Características:** Los bosques aleatorios proporcionan una medida de la importancia de las características, que indica cuánto contribuye cada característica a la precisión del modelo. Esto es útil para la selección de características.

5. **Ventajas y Desventajas:**

   - **Ventajas:** Reducción del sobreajuste en comparación con un solo árbol, manejo automático del sobreajuste, buen rendimiento en conjuntos de datos grandes y complejos.
   - **Desventajas:** Menos interpretables que los árboles de decisión individuales.

22.3.2. Elementos Matemáticos de Árboles de Decisión

1. **Definición:** Un árbol de decisión se representa como una función $T(x)$ que asigna una instancia x a una hoja del árbol. Cada nodo interno j realiza una prueba en una característica $f_j(x)$, y cada rama representa una condición de prueba. La estructura del árbol se define por las funciones indicadoras $I(x, j)$ que indican si la instancia x llega al nodo j.

$$T(x) = \sum_{j=1}^{J} I(x, j) \cdot C_j$$

   Donde J es el número de nodos, C_j es el valor en la hoja correspondiente al nodo j, y $I(x, j)$ es 1 si la instancia x llega al nodo j y 0 de lo contrario.

2. **Proceso de Construcción:** La construcción del árbol implica seleccionar la mejor característica f_j y el umbral t_j en cada nodo j para maximizar la ganancia de información o reducir la impureza de Gini.

$$\text{Ganancia de Información: } Gain(D, j, t) = H(D) - \frac{N_L}{N} H(D_L) - \frac{N_R}{N} H(D_R)$$

$$\text{Impureza de Gini: } Gini(D) = 1 - \sum_{k=1}^{K} \left(\frac{|C_k|}{|D|} \right)^2$$

   Donde D es el conjunto de datos en el nodo, D_L y D_R son los conjuntos de datos en los nodos izquierdo y derecho después de la división, N es el número total de instancias en D, y N_L y N_R son los números de instancias en los nodos izquierdo y derecho.

3. **Ventajas y Desventajas:**

- **Ventajas:** Fácil interpretación, no requiere normalización de datos, manejo natural de características categóricas.
- **Desventajas:** Propenso al sobreajuste, especialmente en conjuntos de datos pequeños y ruidosos.

22.3.3. Bosques Aleatorios

1. **Definición:** Un bosque aleatorio es una colección de B árboles de decisión $T_b(x)$, donde cada árbol se entrena en un subconjunto aleatorio de los datos de entrenamiento. La predicción se obtiene promediando (en regresión) o votando (en clasificación) las predicciones individuales de los árboles.

$$\text{Predicción del Bosque: } \hat{Y}(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$$

2. **Proceso de Construcción:** Cada árbol en el bosque se construye utilizando bagging, que consiste en seleccionar aleatoriamente un subconjunto de las instancias de entrenamiento con reemplazo. Además, en cada división de nodo, se selecciona un subconjunto aleatorio de características.

$$\text{Bagging: } D_b = \{(x_i, y_i)\} \text{ con } i \sim \text{Uniforme}(1, N)$$

$$\text{Características Aleatorias: } f_j \text{ con } j \sim \text{Uniforme}(1, P)$$

3. **Importancia de Características:** La importancia de la característica f_j se mide mediante la disminución promedio en la ganancia de impureza o la reducción en el error cuadrático medio cuando se utiliza f_j para dividir los nodos a lo largo de todos los árboles.

$$\text{Importancia de } f_j = \frac{1}{B}\sum_{b=1}^{B} \sum_{j \text{ en } T_b} \text{Importancia de } f_j \text{ en } T_b$$

22.4. Redes Neuronales

23. Aprendizaje No Supervisado

23.1. K-Means y Clustering Jerárquico

23.2. Análisis de Componentes Principales (PCA)

23.3. Algoritmos de Asociación

23.4. Mapas Autoorganizados (SOM)

24. Evaluación de Modelos y Métricas

24.1. Precisión, Sensibilidad, Especificidad

24.2. Curvas ROC y Área Bajo la Curva (AUC-ROC)

24.3. Matriz de Confusión

24.4. Validación Cruzada

25. Preprocesamiento de Datos

25.1. Normalización y Estandarización

25.2. Manejo de Datos Faltantes

25.3. Ingeniería de Características

25.4. Selección de Características

26. Optimización de Modelos

26.1. Hiperparámetros y Búsqueda en Cuadrícula

26.2. Optimización Bayesiana

26.3. Regularización

26.4. Redes Neuronales Convolucionales (CNN) y Recurrentes (RNN)

27. Aprendizaje por Refuerzo

27.1. Q-Learning

27.2. Algoritmos de Políticas

27.3. Exploración y Explotación

27.4. Funciones de Valor

28. Ética en el Machine Learning

28.1. Sesgo y Equidad

28.2. Transparencia y Explicabilidad

28.3. Privacidad y Seguridad

28.4. Responsabilidad en el Despliegue de Modelos

29. Redes Neuronales Profundas

29.1. Perceptrones Multicapa (MLP)

29.2. Funciones de Activación: ReLU, Sigmoid, Tanh

29.3. Backpropagation

29.4. Regularización en Redes Neuronales

30. Redes Neuronales Convolucionales (CNN)

30.1. Convolutional Layers

30.2. Pooling Layers

30.3. Transfer Learning con CNN

30.4. Aplicaciones en Visión por Computadora

31. Redes Neuronales Recurrentes (RNN)

31.1. Arquitecturas de RNN

31.2. Long Short-Term Memory (LSTM)

31.3. Gated Recurrent Unit (GRU)

31.4. Aplicaciones en Procesamiento de Lenguaje Natural

32. Redes Generativas

32.1. Generative Adversarial Networks (GAN)

32.2. Variational Autoencoders (VAE)

32.3. Aplicaciones en Generación de Imágenes y Texto

33. Transferencia de Aprendizaje en Deep Learning

33.1. Fine-Tuning de Modelos Preentrenados

33.2. Domain Adaptation

33.3. Modelos Preentrenados como BERT, GPT

34. Técnicas Avanzadas

34.1. Normalización por Lotes (Batch Normalization)

34.2. Dropout

34.3. Redes Siamesas

34.4. Redes Neuronales Adversarias Condicionales (cGAN)

35. Herramientas y Frameworks

35.1. TensorFlow

35.2. PyTorch

35.3. Keras

35.4. TensorBoard para Visualización

36. Algoritmos de Aprendizaje Supervisado

36.1. Regresión Lineal y Logística

36.1.1. Regresión Lineal

La regresión lineal es un modelo que busca modelar la relación lineal entre una variable dependiente Y y una o más variables independientes X. El modelo se define como:

$$Y = \beta_0 + \beta_1 X + \varepsilon$$

donde:

- Y es la variable dependiente que queremos predecir.
- X es la variable independiente que utilizamos para la predicción.
- β_0 es la ordenada al origen, que representa el valor de Y cuando X es cero.
- β_1 es la pendiente de la recta, que indica cuánto cambia Y por un cambio unitario en X.
- ε es el término de error que captura la variabilidad no explicada por el modelo.

El objetivo es encontrar los valores de β_0 y β_1 que minimizan la suma de los cuadrados de los errores ε:

$$\min_{\beta_0,\beta_1} \sum_{i=1}^{n} (Y_i - (\beta_0 + \beta_1 X_i))^2$$

Esto se puede hacer utilizando técnicas como el método de mínimos cuadrados.

36.1.2. Regresión Logística

La regresión logística es un modelo utilizado para problemas de clasificación binaria. Se emplea la función logística, también conocida como la función sigmoide, para transformar la salida de la regresión lineal en un valor entre 0 y 1, que se interpreta como la probabilidad de pertenecer a la clase positiva. La función sigmoide está definida como:

$$P(Y = 1) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$

donde:

- $P(Y = 1)$ es la probabilidad de pertenecer a la clase positiva.
- e es la base del logaritmo natural.

El modelo de regresión logística busca encontrar los valores de β_0 y β_1 que maximizan la función de verosimilitud. La función de verosimilitud es el producto de las probabilidades condicionales de observar las etiquetas de clase dados los valores de X:

$$\max_{\beta_0,\beta_1} \mathcal{L}(\beta_0, \beta_1) = \prod_{i=1}^{n} P(Y_i)^{y_i} \cdot (1 - P(Y_i))^{1-y_i}$$

donde:

- $\mathcal{L}(\beta_0, \beta_1)$ es la función de verosimilitud.
- y_i es la etiqueta de la clase para la observación i.

La regresión logística se ajusta típicamente maximizando la log-verosimilitud para obtener estimaciones de β_0 y β_1.

36.1.3. Implementacion

```
# Ejemplo de Regresión Lineal
set.seed(123)
# Crear datos de ejemplo
X <- rnorm(100)
Y <- 2 * X + rnorm(100)

# Ajustar el modelo de regresión lineal
modelo_lineal <- lm(Y ~ X)

# Imprimir resultados
summary(modelo_lineal)

# Graficar el modelo
plot(X, Y, main = "Regresión Lineal", xlab = "X", ylab = "Y")
abline(modelo_lineal, col = "red")

# Ejemplo de Regresión Logística
set.seed(123)
# Crear datos de ejemplo para clasificación binaria
X <- rnorm(100)
probabilidades <- exp(2 * X) / (1 + exp(2 * X))
Y_binario <- rbinom(100, 1, probabilidades)

# Ajustar el modelo de regresión logística
modelo_logistico <- glm(Y_binario ~ X, family = binomial)

# Imprimir resultados
summary(modelo_logistico)
```

```
# Graficar el modelo
plot(X, Y_binario, main = "Regresión Logística",
         xlab = "X", ylab = "Y", col = Y_binario + 1)
curve(predict(modelo_logistico,
           data.frame(X = x), type = "response"),
           add = TRUE, col = "red")
```

36.2. Máquinas de Soporte Vectorial (SVM)

1. **Hiperplano:** En un espacio de características n-dimensional, un hiperplano es un subespacio de dimensión $n-1$. Para un problema de clasificación binaria, un hiperplano divide el espacio en dos regiones, asignando puntos a una clase u otra.

2. **Margen:** El margen es la distancia perpendicular desde el hiperplano a los puntos más cercanos de cada clase. SVM busca el hiperplano que maximiza este margen, lo que se traduce en una mayor robustez y generalización del modelo.

3. **Vectores de Soporte:** Estos son los puntos de datos más cercanos al hiperplano y tienen un papel crucial en la definición del margen. Cambiar estos vectores de soporte afecta directamente al modelo, y son los únicos puntos que importan para la determinación del hiperplano.

4. **Función de Decisión:** La función de decisión de SVM es el hiperplano que se utiliza para clasificar nuevos puntos de datos. Dada una entrada, la función de decisión evalúa de qué lado del hiperplano cae el punto y asigna la etiqueta correspondiente.

5. **Kernel Trick:** SVM puede manejar eficientemente datos no lineales mediante el uso de funciones de kernel. Estas funciones transforman el espacio de características original en uno de mayor dimensión, permitiendo así que los datos sean separados de manera no lineal en el espacio transformado.

6. **Parámetros de SVM:**

   - **C (Parámetro de Regularización):** Controla el equilibrio entre tener un margen más amplio y clasificar correctamente los puntos de entrenamiento.
   - **Kernel:** Define la función de kernel utilizada (lineal, polinómica, radial, etc.).
   - **Gamma (para kernels no lineales):** Controla el alcance de influencia de un solo punto de datos en la decisión.

7. **Proceso de Entrenamiento:** Dado un conjunto de datos de entrenamiento etiquetado, SVM busca el hiperplano óptimo que maximiza el margen entre las clases. Esto se realiza a través de técnicas de optimización cuadrática.

8. **SVM para Regresión:** Además de la clasificación, SVM se puede utilizar para problemas de regresión. En este caso, el objetivo es ajustar un hiperplano de modo que contenga la mayor cantidad posible de puntos dentro de un margen predefinido.

9. **Ventajas y Desventajas:**
    - **Ventajas:** Efectivo en espacios de alta dimensión, eficaz en conjuntos de datos pequeños y versátil gracias al kernel trick.
    - **Desventajas:** Sensible a la escala de las características, puede ser computacionalmente costoso para grandes conjuntos de datos y requiere la elección cuidadosa de parámetros.
10. **Aplicaciones:** SVM se utiliza en una variedad de campos, como reconocimiento de escritura, clasificación de imágenes, diagnóstico médico, entre otros.

36.2.1. Elementos Matemáticos de las SVM

1. **Hiperplano:** Un hiperplano se define como $w \cdot x - b = 0$, donde w es el vector de pesos, x es el vector de entrada, y b es el sesgo.
2. **Margen:** El margen M entre un hiperplano y un punto x_i se define como $M = \frac{1}{\|w\|}|w \cdot x_i - b|$.
3. **Vectores de Soporte:** Los vectores de soporte son los puntos x_i que cumplen la condición $|w \cdot x_i - b| = 1/\|w\|$.
4. **Función de Decisión:** La función de decisión es $f(x) = w \cdot x - b$. Si $f(x) > 0$, el punto x se clasifica como clase 1; si $f(x) < 0$, se clasifica como clase -1.
5. **Kernel Trick:** La función de kernel $K(x, x')$ representa el producto escalar en un espacio de características de mayor dimensión. Ejemplos comunes incluyen el kernel lineal ($K(x, x') = x \cdot x'$), kernel polinómico ($K(x, x') = (x \cdot x' + 1)^d$), y kernel radial ($K(x, x') = \exp(-\gamma\|x - x'\|^2)$).
6. **Parámetros de SVM:**
    - C (Parámetro de Regularización): Se introduce en la función de pérdida para controlar el equilibrio entre tener un margen más amplio y clasificar correctamente los puntos de entrenamiento.
    - w (Vector de Pesos): Aprende durante el entrenamiento y define la orientación del hiperplano.
    - b (Sesgo): Parámetro de ajuste del hiperplano.
    - γ (para kernels no lineales): Controla el alcance de influencia de un solo punto de datos en la decisión.
7. **Proceso de Entrenamiento:** El proceso de entrenamiento implica la minimización de la función de pérdida, que incluye el término de regularización $C\|w\|^2$ y la función de pérdida hinge.
8. **SVM para Regresión:** Para regresión, el objetivo es ajustar un hiperplano de modo que $|w \cdot x_i - b| \leq \epsilon$ para puntos de entrenamiento x_i.

9. **Ventajas y Desventajas:**
    - Ventajas: Efectivo en espacios de alta dimensión, eficaz en conjuntos de datos pequeños y versátil gracias al kernel trick.
    - Desventajas: Sensible a la escala de las características, puede ser computacionalmente costoso para grandes conjuntos de datos y requiere la elección cuidadosa de parámetros.
10. **Aplicaciones:** SVM se aplica en una variedad de problemas, como reconocimiento de escritura, clasificación de imágenes y diagnóstico médico.

36.3. Árboles de Decisión y Bosques Aleatorios

1. **Definición:** Un árbol de decisión es una estructura jerárquica en forma de árbol que se utiliza para representar decisiones y sus posibles consecuencias. Cada nodo interno del árbol representa una prueba en una característica, cada rama representa un resultado posible de la prueba, y cada hoja representa un resultado final o una decisión.
2. **Proceso de Construcción:** El árbol se construye de manera recursiva. En cada paso, se elige la mejor característica para dividir el conjunto de datos en función de algún criterio, como la ganancia de información o la impureza de Gini. Este proceso se repite hasta que se alcanza algún criterio de parada, como la profundidad máxima del árbol o un número mínimo de puntos en una hoja.
3. **Criterios de División:** Los criterios comunes para la división incluyen:
    - **Ganancia de Información:** Mide cuánta información nueva proporciona una característica.
    - **Impureza de Gini:** Mide la probabilidad de clasificar incorrectamente un elemento si es etiquetado aleatoriamente.
4. **Ventajas y Desventajas:**
    - **Ventajas:** Fácil interpretación, no requiere normalización de datos, manejo natural de características categóricas.
    - **Desventajas:** Propenso al sobreajuste, especialmente en conjuntos de datos pequeños y ruidosos.

36.3.1. Bosques Aleatorios

1. **Definición:** Un bosque aleatorio es una colección de árboles de decisión entrenados en subconjuntos aleatorios del conjunto de datos y utilizando técnicas de agregación para mejorar la precisión y controlar el sobreajuste.
2. **Proceso de Construcción:** Se crean múltiples árboles de decisión utilizando diferentes subconjuntos del conjunto de datos de entrenamiento y características aleatorias en cada división. Luego, las predicciones de cada árbol se promedian (en regresión) o se votan (en clasificación) para obtener la predicción final del bosque.

3. **Técnica de Bagging:** La construcción de árboles en subconjuntos aleatorios del conjunto de datos se conoce como bagging (bootstrap aggregating). Esto ayuda a reducir la varianza y evitar el sobreajuste al promediar los errores.

4. **Importancia de Características:** Los bosques aleatorios proporcionan una medida de la importancia de las características, que indica cuánto contribuye cada característica a la precisión del modelo. Esto es útil para la selección de características.

5. **Ventajas y Desventajas:**

   - **Ventajas:** Reducción del sobreajuste en comparación con un solo árbol, manejo automático del sobreajuste, buen rendimiento en conjuntos de datos grandes y complejos.
   - **Desventajas:** Menos interpretables que los árboles de decisión individuales.

36.3.2. Elementos Matemáticos de Árboles de Decisión

1. **Definición:** Un árbol de decisión se representa como una función $T(x)$ que asigna una instancia x a una hoja del árbol. Cada nodo interno j realiza una prueba en una característica $f_j(x)$, y cada rama representa una condición de prueba. La estructura del árbol se define por las funciones indicadoras $I(x, j)$ que indican si la instancia x llega al nodo j.

$$T(x) = \sum_{j=1}^{J} I(x, j) \cdot C_j$$

   Donde J es el número de nodos, C_j es el valor en la hoja correspondiente al nodo j, y $I(x, j)$ es 1 si la instancia x llega al nodo j y 0 de lo contrario.

2. **Proceso de Construcción:** La construcción del árbol implica seleccionar la mejor característica f_j y el umbral t_j en cada nodo j para maximizar la ganancia de información o reducir la impureza de Gini.

$$\text{Ganancia de Información: } Gain(D, j, t) = H(D) - \frac{N_L}{N} H(D_L) - \frac{N_R}{N} H(D_R)$$

$$\text{Impureza de Gini: } Gini(D) = 1 - \sum_{k=1}^{K} \left(\frac{|C_k|}{|D|} \right)^2$$

   Donde D es el conjunto de datos en el nodo, D_L y D_R son los conjuntos de datos en los nodos izquierdo y derecho después de la división, N es el número total de instancias en D, y N_L y N_R son los números de instancias en los nodos izquierdo y derecho.

3. **Ventajas y Desventajas:**

- **Ventajas:** Fácil interpretación, no requiere normalización de datos, manejo natural de características categóricas.
- **Desventajas:** Propenso al sobreajuste, especialmente en conjuntos de datos pequeños y ruidosos.

36.3.3. Bosques Aleatorios

1. **Definición:** Un bosque aleatorio es una colección de B árboles de decisión $T_b(x)$, donde cada árbol se entrena en un subconjunto aleatorio de los datos de entrenamiento. La predicción se obtiene promediando (en regresión) o votando (en clasificación) las predicciones individuales de los árboles.

$$\text{Predicción del Bosque: } \hat{Y}(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$$

2. **Proceso de Construcción:** Cada árbol en el bosque se construye utilizando bagging, que consiste en seleccionar aleatoriamente un subconjunto de las instancias de entrenamiento con reemplazo. Además, en cada división de nodo, se selecciona un subconjunto aleatorio de características.

$$\text{Bagging: } D_b = \{(x_i, y_i)\} \text{ con } i \sim \text{Uniforme}(1, N)$$

$$\text{Características Aleatorias: } f_j \text{ con } j \sim \text{Uniforme}(1, P)$$

3. **Importancia de Características:** La importancia de la característica f_j se mide mediante la disminución promedio en la ganancia de impureza o la reducción en el error cuadrático medio cuando se utiliza f_j para dividir los nodos a lo largo de todos los árboles.

$$\text{Importancia de } f_j = \frac{1}{B}\sum_{b=1}^{B} \sum_{j \text{ en } T_b} \text{Importancia de } f_j \text{ en } T_b$$

36.4. Redes Neuronales

37. Aprendizaje No Supervisado

37.1. K-Means y Clustering Jerárquico

37.2. Análisis de Componentes Principales (PCA)

37.3. Algoritmos de Asociación

37.4. Mapas Autoorganizados (SOM)

38. Evaluación de Modelos y Métricas

38.1. Precisión, Sensibilidad, Especificidad

38.2. Curvas ROC y Área Bajo la Curva (AUC-ROC)

38.3. Matriz de Confusión

38.4. Validación Cruzada

39. Preprocesamiento de Datos

39.1. Normalización y Estandarización

39.2. Manejo de Datos Faltantes

39.3. Ingeniería de Características

39.4. Selección de Características

40. Optimización de Modelos

40.1. Hiperparámetros y Búsqueda en Cuadrícula

40.2. Optimización Bayesiana

40.3. Regularización

40.4. Redes Neuronales Convolucionales (CNN) y Recurrentes (RNN)

41. Aprendizaje por Refuerzo

41.1. Q-Learning

41.2. Algoritmos de Políticas

41.3. Exploración y Explotación

41.4. Funciones de Valor

42. Ética en el Machine Learning

42.1. Sesgo y Equidad

42.2. Transparencia y Explicabilidad

42.3. Privacidad y Seguridad

42.4. Responsabilidad en el Despliegue de Modelos

43. Redes Neuronales Profundas

43.1. Perceptrones Multicapa (MLP)

43.2. Funciones de Activación: ReLU, Sigmoid, Tanh

43.3. Backpropagation

43.4. Regularización en Redes Neuronales

44. Redes Neuronales Convolucionales (CNN)

44.1. Convolutional Layers

44.2. Pooling Layers

44.3. Transfer Learning con CNN

44.4. Aplicaciones en Visión por Computadora

45. Redes Neuronales Recurrentes (RNN)

45.1. Arquitecturas de RNN

45.2. Long Short-Term Memory (LSTM)

45.3. Gated Recurrent Unit (GRU)

45.4. Aplicaciones en Procesamiento de Lenguaje Natural

46. Redes Generativas

46.1. Generative Adversarial Networks (GAN)

46.2. Variational Autoencoders (VAE)

46.3. Aplicaciones en Generación de Imágenes y Texto

47. Transferencia de Aprendizaje en Deep Learning

47.1. Fine-Tuning de Modelos Preentrenados

47.2. Domain Adaptation

47.3. Modelos Preentrenados como BERT, GPT

48. Técnicas Avanzadas

48.1. Normalización por Lotes (Batch Normalization)

48.2. Dropout

48.3. Redes Siamesas

48.4. Redes Neuronales Adversarias Condicionales (cGAN)

49. Herramientas y Frameworks

49.1. TensorFlow

49.2. PyTorch

49.3. Keras

49.4. TensorBoard para Visualización

50. Introducción y antecedentes

50.1. Artículo 1: Machine Learning in Enzyme Engineering

Título: Machine Learning in Enzyme Engineering, Stanislav Mazurenko, Zbynek Prokop, and Jiri Damborsky [**?**]

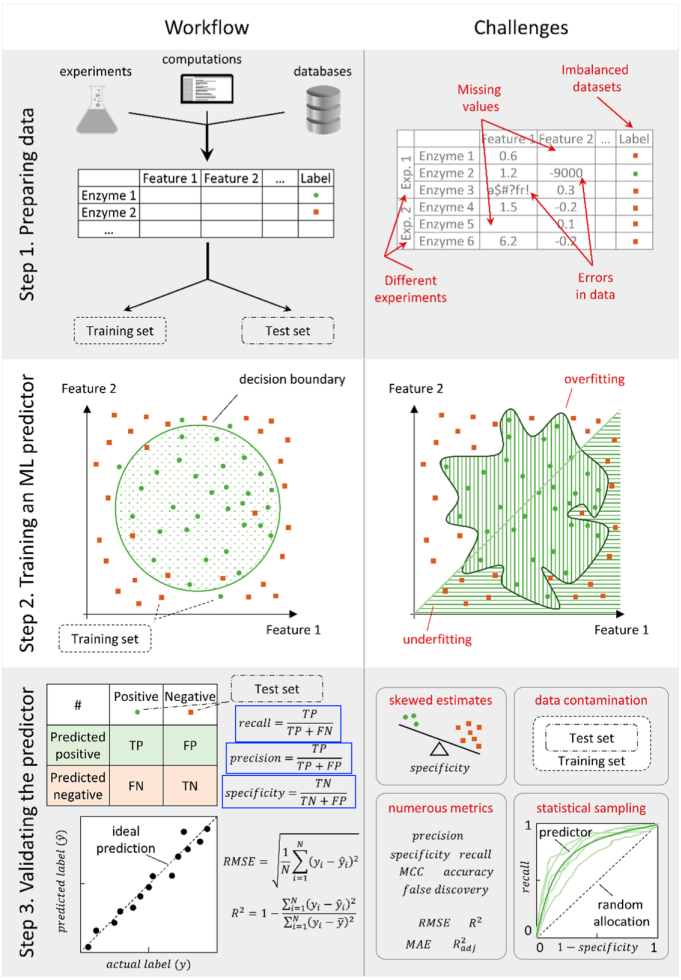
- Enzyme engineering is the process of customizing new biocatalysts with improved properties by altering their constituting sequences of amino acids.
- Multiple ML algorithms have already been applied to enzyme engineering. Some notable examples include random forests used to predict protein solubility [**?**], support vector machines [**?**, **?**] and decision trees [**?**] to predict enzyme stability changes upon mutations, K-nearest-neighbor classifiers to predict enzyme function[**?**] and mechanisms,[**?**] and various scoring and clustering algorithms for rapid functional sequence annotation [**?**, **?**]. The main attractiveness of ML in enzyme engineering stems from its generalizability: once it is trained on the known input, called a training set, an ML algorithm can potentially make predictions about new variants almost instantly.
- The aim of this Perspective is, therefore, to highlight recent advances in data collection and algorithm implementation for ML in enzyme engineering.

50.2. The essence of Machine Learning

La esencia de la mayoría de los algoritmos de Machine Learning (ML) es encontrar patrones en los datos disponibles, datos que consisten en varios descriptores o características, por ejemplo secuencias de encimas, sus estructuras secundarias y terciarias, substiruciones, etc. El nímero de características usualmente varían de decenas a miles lo que convierte el problema en uno de alta dimensión.

Los principales tipos de Machine Learning son: Aprendizaje Supervisado y Aprendizaja No-Supervisado. En el aprensizaje no supervisado el objetivo es disminuir la alta dimensionalidad de los datos en uno de menor dimensión, o el de encontrar clústers en los datos. En el aprendizaje supervisado varias propiedades objetivo tales como actividad o estabilidad de enzimas, y el objetivo es diseñar un predictor que regrese etiquetas para datos no vistos considerando sus descriptores, utilizando el conjunto de datos etiquetado como datos de entrenamiento.

Nota 6. *Step 1: the data are usually turned into a table format and split into the training and test parts. Any errors, biases, or imbalances will be translated to the predictor's performance and, hence, must be accounted for. Step 2: the predictor is trained on the training data set. For example, a decision boundary is derived that allows classifying future input based on whether data points are inside or outside the boundary. This is a balancing act between two extremes: explaining noise rather than fundamental dependencies (overfitting) or failure to account for complex dependencies in the data (underfitting). Step 3: the performance of the predictor is evaluated based on the test data set. For example, true and false positives and negatives and the associated measures are calculated or the root mean square error (RMSE)*



ht!

Figura 1: Schematic workflow of constructing an ML predictor and associated challenges.

is calculated for continuous labels. The random nature of the initial data split as well as data imbalances might skew the evaluation, and numerous metrics used for evaluation vary in their robustness to different data skews. Even partial inclusion of the test set at any stage of ML predictor training is called data contamination and usually invalidates the final evaluation.

La etapa que más tiempo consume es la de recolección de datos y su preparación para alimentar el algoritmo de ML, entonces los datos son introducidos en el subconjunto de entrenamiento, el resultado se utiliza para meujorar los parámetros del predictor de ML, mientras que el segundo se utiliza para la evaluación.

Nota 7.

- *En problemas de clasificación con etiquetas binarias o etiquetas con una cantidad finita de opciones, la evaluación usualmente se realiza por medio de la matriz de confusión: el número de verdaderos/falsos positivos y negativos.*

	Positivo	*Negativo*
Predecido Positivo	*TP*	*FP*
Predicido Negativo	*FN*	*TN*

- *Para problemas de regresión con etiquetas de valores continuos usualmente se calcula la raíz del error cuadrático medio*

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2} \tag{1}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y})^2}{\sum_{i=1}^{N}(y_i - \overline{y})^2} \tag{2}$$

En cualquiera de los dos casos la evaluación final se lleva a cabo en el conjunto de prueba, el cuál es esencial dado que el último objetivo es obtener el predictor más general en los datos no utilizados para entrenar el algoritmo.

Nota 8. *Las siguientes métricas se utilizan para medir el rendimiento de un modelo en función de su capacidad para predecir correctamente las clases de un conjunto de datos.*

- ***Recall (Recall o Sensibilidad):*** *Conocido como sensibilidad o tasa positiva real, mide la capacidad de un modelo para identificar correctamente todos los ejemplos positivos en un conjunto de datos. Se calcula como el número de verdaderos positivos dividido por la suma de verdaderos positivos y falsos negativos:*

$$Recall = \frac{Verdaderos\ Positivos}{Verdaderos\ Positivos + Falsos\ Negativos} \tag{3}$$

Un recall alto significa que el modelo es bueno para detectar los casos positivos, minimizando los falsos negativos. Es importante en situaciones donde los falsos negativos son costosos o críticos.

- *Precision (Precisión): La precisión mide la capacidad de un modelo para predecir correctamente los casos positivos entre todas las predicciones positivas que realiza. Se calcula como el número de verdaderos positivos dividido por la suma de verdaderos positivos y falsos positivos:*

$$Precision = \frac{Verdaderos\ Positivos}{Verdaderos\ Positivos + Falsos\ Positivos} \tag{4}$$

Una alta precisión significa que el modelo tiene una baja tasa de falsos positivos, es decir, que cuando predice una clase como positiva, es probable que sea correcta. La precisión es importante en situaciones en las que los falsos positivos son costosos o no deseados.

- *Specificity (Especificidad): La especificidad mide la capacidad de un modelo para predecir correctamente los casos negativos entre todas las predicciones negativas que realiza. También se conoce como tasa negativa real. Se calcula como el número de verdaderos negativos dividido por la suma de verdaderos negativos y falsos positivos:*

$$Specificity = \frac{Verdaderos\ Negativos}{Verdaderos\ Negativos + Falsos\ Positivos} \tag{5}$$

Una alta especificidad indica que el modelo es bueno para identificar correctamente los casos negativos, minimizando los falsos positivos. Esto es importante en situaciones en las que los falsos positivos son costosos o problemáticos.

Estas métricas proporcionan una forma más completa de evaluar el rendimiento de un modelo de clasificación que simplemente mirar la precisión general.

En la ingeniería de proteínas, las similitudes en secuencias en ambos subconjuntos de datos deben ser tenidas en cuenta. Si alguna familia de proteínas está sobre representada en el conjunto de prueba, el predictor resultante puede resultar sesgado hacia la identificación de patrones válidos solamente para esta familia. Si algunas secuencias en el conjunto de prueba son muy cercanas al conjunto de entrenamiento, la evaluación final de desempeño dara resultados sobre optimistas.

En el paso 2 de entrenamiento, es posible ajustar el predictor o seleccionar de entre varios predictores, usualmente por medio de validación $k - fold$. En este caso los datos de entrenamiento se subdividen en K subconjuntos y el flujo de trabajo se repite K veces, con cada uno de ellos utilizaods para la evaluación de los $K - 1$ subconjuntos utilizados para entrenar. El reto principal en el paso 2 para cualquiern entrenamiento tipo ML supervisado es evitar el subajuste de los datos (sesgo alto) y el sobre ajuste (varianza grande).

La **subestimación** ocurre cuando un predictor falla en encontrar patrones incluso en los datos de entrenamiento (cuando un modelo lineal simple se utiliza para explicar dependencia dependencias no lineales en los datos). El **sobreajuste** ocurre cuando el desempeño de un predictor disminuye notablemente en los datos de prueba en comparación con los datos de prueba, debido al aprendizaje de demasiado detalle y ruido, en lugar de identificar patrones generales. Tanto el subajuste como el sobreajuste pueden ser debido a la insuficiente calidad de los datos: ruido excesivo, características faltantes o irrelevantes, sesgo en los datos, o

datos dispersos. También pueden ocurrir como consecuencia de una pobre aplicación del algoritmo: excesiva o insuficiente flexibilidad en la selección de los parámetros, protocolo de entrenamiento inapropiado, o contaminación de los datos de entrenamiento con el conjunto de datos de prueba.

50.3. Bases de datos relevantes a Ingeniería de Enzima

50.3.1. The State of the Art in Data Accumulation

Debido a que los algoritmos de ML se basan en los datos, la importancia de la calidad de los mismos utilizados para entrenamiento no puede ser subestimada.

Ejemplos de conjuntos de bases de datos, utilizadas en la ingeniería de enzimas, son secuencias de proteínas y estructuras de proteínas. La estabilidad y solubilidad de las proteínas son dos cualidades que han sido medidas por varias décadas y hasta la fecha. Tareas más desafiante es la anotar las propiedades catalíticas de las enzimas debido a la abundancia de tipos de reacciones, mecanismos, cofactores, amplios rangos de especificidades de sustratos, enancionselectividades y promiscuidades.

Las enzimas son catalizadores biológicos que facilitan una amplia variedad de reacciones químicas en los organismos vivos. Algunas razones por las cuales la anotación de sus propiedades catalíticas es complicada son:

- **Tipos de reacciones diversos:** Las enzimas pueden catalizar una amplia gama de reacciones químicas, incluyendo reacciones de oxidación-reducción, hidrólisis, condensación, isomerización y más. Cada tipo de reacción involucra mecanismos químicos y sustratos diferentes.

- **Mecanismos:** Incluso dentro de un solo tipo de reacción, las enzimas pueden emplear múltiples mecanismos. Comprender el mecanismo específico utilizado por una enzima requiere un conocimiento detallado de la estructura de la enzima y de su sitio activo.

- **Cofactores:** Muchas enzimas requieren cofactores, como iones metálicos o coenzimas, para catalizar reacciones de manera efectiva. Identificar los cofactores necesarios para cada enzima es esencial para la anotación.

- **Condiciones de reacción:** La actividad enzimática puede depender en gran medida de las condiciones ambientales, incluyendo la temperatura, el pH y la fuerza iónica. La anotación de las condiciones óptimas para la actividad enzimática es crucial.

- **Especificidades de sustratos:** Las enzimas pueden ser altamente específicas para ciertos sustratos, reconociéndolos con alta afinidad, mientras que otras son más promiscuas y pueden unirse a una variedad de sustratos. La caracterización de la especificidad de sustratos es compleja.

- **Enantioselectividades:** Algunas enzimas pueden discriminar entre enantiómeros (isómeros de imagen especular) de una molécula, catalizando reacciones con alta selectividad por un enantiómero. La anotación de esta propiedad implica comprender la estereoquímica.

- **Promiscuidades:** Las enzimas pueden exhibir actividades promiscuas, catalizando reacciones diferentes a su función principal. Detectar y caracterizar tales promiscuidades es complicado.

50.3.2. Current Challenges Related to Databases

Si la dependencia que se busca no se encuentra en los datos disponibles, no importa la cantidad de nuevos datos ayudarán a mejorar la calidad del predictor de ML. En el caso de la ingeniería de enzimas se espera que las funciones enzimáticas estén codificadas en las secuencias y así depender en las propiedades físico-químicas de los aminoácidos, de aquí que la cantidad y la calidad de los datos en las bases de datos sean de importancia para diseñar un predictor de ML.

La falta de estándares en los reportes resulta en pérdida de información o valores erróneos para algunos descriptores. A esto hay que agregar la falta de protocolos robustos en los análisis de datos, como por ejemplo aquellos utilizados para ajuste de curvas para determinar las temperaturas de fusión o constantes cinéticas. Otro factor es que los recientes avances vuelven obsoletos resultados previos. La curación manual ayuda mejorar la calidad de los datos, sin embargo no se encuentra excenta de errores de anotación de las funciones de las proteínas y errores de propagación a partir de resultados previamente refutados.

Este tipo de procedimientos, verificación manual, puede incluir la limpieza o formato de datos para que sean amigables con ML. Uno de los principios más populares es **FAIR**, por sus siglas en inglés, Findable, Accesible, Interoperables y Reutilizables, debería de facilitar a las computadoras para que de manera automática pueda encontrar y utilizar los datos. Para las enzimas la guía estándar para reportar datos de enzimas (STRENDA) debería de aumentar la calidad de los datos, especialmente en bases de datos (bdd) heterogéneas recopiladas de diversas fuentes.

El desarrollo de nuevos predictores de ML ha incrementado considerablemente la demanda de mejora de las bdd existentes, así como la generación de nuevas bdd uniformes y representativas de mayor calidad.

Existen varias nuevas técnicas emergentes tales como

i Secuenciación de nueva generación.

ii Clasificación de células activadas por fluorescencia.

iii Exploración mutacional profunda, y

iv Microfluidos

50.3.3. Emerging Methods for High-Throughput Data Collection

Avances tecnológicos hacia la miniaturización, automatización y paralelización han generado tecnologías eficientes de nuevos métodos de investigación experimental con capacidades incomparables. Secuenciación de nueva generación (NGS) ha revolucionado la investigación genómica, habilitado el acceso a datos moleculares fundamentales y revelado firmas genómicas y transcriptomicas [**?**, **?**].

La capacidad de secuencias en el rango de gigabases por ejecución del instrumento permite secuencias el genoma humano en su totalidad en tan sólo un día.

Múltiples instrumentos comerciales de segunda generación disponibles ofrecen mayor capacidad y precisión. Métodos de tercera generación recientemente introducidos (lectura larga) que emplean secuenciación en tiempo real de moléculas [**?**] o secuenciación de nanoporos [**?**] resuelve las limitaciones de lecturas cortas, como el sesgo de GC o mapear elementos repetitivos

Mientas el avance de la tecnología de secuenciación proporciona una gran cantidad de secuencias de datos, para la mayoría de estas entradas, las anotaciones estructurales y anotaciones funcionales aún están perdidas.

Como siguiente paso, se está centrando en el desarrollo de nuevos métodos experimentales efectivos para recopilar información funcional y estructural.

La clasificación de células activadas por fluorescencia (FACS) es una tecnología ampliamente disponible que permite el cribado de hasta 108 variantes de enzimas por día. La FACS requiere que los sustratos fluorogénicos estén atrapados dentro o en la superficie de la célula para vincular el genotipo y el fenotipo. Alternativamente, se utiliza la clasificación de enzimas encapsuladas junto con su ADN codificador y un sustrato fluorogénico en perlas de hidrogel.

Cuando se combinan con la secuenciación de próxima generación, los ensayos de alto rendimiento representan una estrategia poderosa para analizar de manera integral las relaciones entre secuencia y función en las enzimas [**?**, **?**]. Este enfoque, llamado escaneo mutacional profundo (DMS por sus siglas en inglés), vincula el genotipo con el fenotipo sin necesidad de procesos laboriosos que involucren la purificación y caracterización de proteínas. Durante el proceso, se sintetiza una gran biblioteca de secuencias mutantes, seguida de la selección de fenotipos expresados. Luego, la secuenciación de la biblioteca antes y después de la selección cuantifica la aptitud de cada mutante. De esta manera, el DMS proporciona un método rápido y sencillo para inferir los determinantes de la secuencia de la estabilidad y la función de las proteínas[**?**, **?**, **?**]. El DMS se ha utilizado como una estrategia experimental alternativa para la determinación de la estructura de las proteínas.

50.4. MACHINE LEARNING APPLICATIONS TO ENZYME ENGINEERING

- Despite being a relatively new field of study, machine Learning for enzyme engineering has already been applied for several challenging predictions. First consider predictors aimed at elucidating the structure function relationships crucial for enzymes on both sides:
    - predicting the structure for a known sequence, and
    - predicting the catalytic activity or substrate specificity for a known sequence-structure.
    - solubility and stability, from the point of view of amino acid substitutions,
- The protein structure prediction is one of the longest-standing challenges in biochemistry, as the number of resolved structures is dramatically lagging behind the number

of known sequences.

- Over 145000 structures have been released in the Protein Data Bank, but this is still nowhere near over 215 million publicly available protein sequences[**?**].

- Nevertheless, even despite a relatively small data set size in comparison to millions of data points usually available for this method, deep neural networks showed most the notable results in the latest biennial assessment of protein structure prediction methods, CASP13.

  Nota 9. *The AlphaFold network was trained on the PDB entries to predict the distances between C-beta atoms of residues using multiple sequence alignments[**?**] and received the highest score at the competition. Out of 124 targets, around two-thirds of AlphaFold predictions had a GDT_TS score above 50, which is indicative of a topologically correct structure [**?**].*

- Apart from predicting protein structures, predicting catalytic activities is another active field of research currently.

- Computational methods for the protein function prediction range from sequence-to structure-based and from gene-to genome- and interactome-based[**?**].

- Several initiatives similar to the CASP competition have already been proposed to address the functional annotation of enzymes, namely Enzyme Function Initiative (EFI), the Computational Bridges to Experiments initiative (COMBREX), and the Critical Assessment of Function Annotation community-driven experiment (CAFA).

- Successful attempts to apply ML to assign enzyme EC numbers using predicted 3D structures [**?**] or exploiting sequence similarities [**?**] have already been made.

- Recently, deep learning was also applied to predict EC numbers on the basis of a protein sequence using both sequence-length-dependent features, such as raw sequence one-hot encoding, and sequence-length-independent features, such as functional domain encoding [**?**].

- The former type of features introduced non uniformity in feature dimensionality, and the authors presented a framework to perform simultaneously dimensionality uniformization, feature selection, and classification model training.

- The large data sets of enzyme structures and activities accumulated to date already allow using deep learning in the engineering of catalytic activity.

- A more precise functional prediction is possible by restricting ML training to a particular family of enzymes, which comes at the cost of much smaller data sets available for training. This problem may be tackled by applying high-throughput data collection methods mentioned before. The authors of the recently released GT-predict [**?**] selected for their analysis the glycosyltransferase superfamily 1, a group of enzymes with highly diverse substrates.

Nota 10. *Data from the label-free mass spectroscopy-based assay of 91 substrates and 54 enzymes derived from the plant Arabidopsis thaliana were used for functional prediction. The authors trained sequence-based decision trees, systematically varying combinations of physicochemical properties, e.g. log P, molecular area, and number/type of nuclephilic groups, and structural information, e.g. scaffold type and functional groups. The resulting predictor was successfully tested on four individually selected gene sequences as well as two complete families of enzymes from four different organisms, which highlights the tremendous potential of training ML predictors on the newly acquired data from high-throughput data collection methods.*

- In their recent paper [**?**] Han and coauthors considered seven different binary and continuous ML algorithms: logistic regression, decision tree, support vector machines, Naive Bayes, conditional random forest, XGBoost, and artificial neural networks.
- The authors attempted to use generative adversarial networks to synthesize more data. This is a pair of two neural networks competing against each other: one learns to generate artifficial examples and the other to distinguish them from real data.
- Another point of view on protein solubility prediction is studying the effects of individual mutations. The recent successes in the application of deep mutational scanning to collect the data on protein solubility changes upon mutations[**?**] are likely to promote the development of more sophisticated ML-based protein solubility predictors in the nearest future.
- Predicting the effects of amino acid substitutions is not only limited to solubility: stability, substrate specificity, catalytic activity, and enantioselectivity can also be targeted if sufficient data are available.
- Protein stability predictors are perhaps those with the most abundant data sets of this type available for ML training
- The authors also presented a random forest classiffier trained using 1106 features from the following groups: experimental conditions, conservation and coevolution scores for mutated positions, amino acid substitutions and their physicochemical properties, neighborhood features for 11 positions before and after substitution sites, and thermodynamic sequence-based features extracted from ProtDCal [**?**].
- PON-tstab is a three-class predictor (stability increasing, decreasing, unchanged) and achieved the correct prediction ratio of around 0.5 versus the value 0.33 for a random predictor. This implies that, even with a data set of higher quality, predicting protein stability remains an extremely challenging task [**?**].
- Another intriguing application of ML in protein engineering is to design smart combinatorial libraries for directed protein evolution[**?**]. This has the potential to both reduce the experimental effort and improve the exploration of the sequence space by mutating multiple positions simultaneously. Moreover, it can approximate the empirical fitness landscape to suggest a refined set of variants for the next round of screening.

- Wu et al[**?**] used ML assisted directed evolution to engineer an enzyme for a new stereodivergent carbon-silicon bond formation. The authors selected the reaction of phenyldimethyl silane with ethyl 2-diazopropanoate catalyzed by a putative nitric oxide dioxygenase from Rhodothermus marinus. They tested a variety of ML algorithms such as linear and kernel models, shallow neural networks, and ensemble methods to improve the enzyme enantioselectivity.

50.4.1. Current Challenges Related to ML-Aided

- One of the main challenges in applications of ML to enzyme engineering stems from the intrinsic multidisciplinarity of the approach. Biochemists, molecular biologists, mathematicians, and computer scientists have to find a common language to clarify goals, carry out rigorous analysis and training, and avoid common pitfalls, wrongful usage of methods, and misinterpretations.

- The No Free Lunch theorem [**?**] claims that no single ML method is superior to others a priori [**?**];

- A thorough understanding of the data types to be used and problems to be solved is essential in the development of efficient predictors. The current shift toward new and more complex ML methods, namely aggregating several algorithms into hybrid meta-predictors, hyperparameter optimization with many training subcycles, feature learning, and the fusion of ML-based and classical bioinformatics tools in a single predictor, will further challenge the crosstalk between disciplines necessary for the development of efficient and robust predictors in enzyme engineering.

- With the continuous growth of ML applications in enzyme engineering, the need for robust comparison of various predictors is of growing importance. This comparison is mainly obstructed by the lack of both standardized protocols for comparison and new data sets for testing.

- The lack of benchmark data sets, discrepancies in the performance measurements used, inaccurate or insufficient disclosure in publications, and the difficulty in finding reviewers with sufficiently broad expertise [**?**] are among the most pressing issues.

- Researchers working on some applications with a long track record in bioinformatics, such as protein structure or function predictions, have already established several platforms that can be used for comparison of the ML predictors, i.e. CASP, CAFA, EFI, and COMBREX.

- Other applications have yet to see similar initiatives as at least three key ingredients are necessary:

  i a sufficiently large community of researchers working on development of such applications,

  ii a sufficient amount of new high-quality data being collected regularly, and

iii a leader that will take on responsibility and invest time and effort into coordinating this activity.

- Few papers go beyond simple ROC analysis: e.g., resample cross-validation to estimate its statistical significance, explore the reasons for weak predictions, and analyze learning curves. Why does a particular predictor have a better performance? What features are critical for the performance of a predictor on a global scale? What ranges for feature values and what parts of the feature space are most critical for a particular data point to be classiffied correctly?

50.5. Emerging Trends in ML-Based Methods for Enzyme Engineering

- With the accumulation of more data by virtue of the emerging high-throughput experimental methods, the development of benchmark data sets and unified performance measurements is only a matter of time.

- Recently, an intriguing algorithm based on semisupervised learning has been presented to allow benchmarking in five different prediction tasks related to protein engineering, including secondary structure, fluorescence landscape, and stability landscape predictions [**?**]

- As the data generation is streamlined, a data set from a single experiment is starting to have the size large enough for training ML algorithms to guide the design of future experiments, as was the case in the development of stereodivergent carbon silicon bond formation[**?**] and the application of Gaussian processes to the directed evolution of cytochromes[**?**].

- The increase in the available data will prompt more extensive use of deep neural networks. Sophisticated neural network architectures, such as recurrent or graph-based neural networks, simultaneous training of several types of predictors (multitasking), combining structurally different input data (multimodal design), ML-based modeling of data sets (generative models), and retraining predictors used in one area by new data from another area (transfer learning) have only recently been applied in genomics[**?**]

- Several attempts have recently been made to apply some of those advanced techniques to proteins: using generative models to create soluble and functional malate dehydrogenase variants[**?**] or predict mutational effects with high correlation with those actually observed in 42 high-throughput deep mutational scanning experiments[**?**]

- More data will also allow improving the existing methods, i.e. learning the optimal architecture of a predictor from the data (hyperparameter optimization)[**?**], smart aggregation of several predictions from multiple methods[**?**], and introducing robust conffidence scores for predictions[**?**]. In enzyme engineering, this new level of algorithmic complexity will further save time and resources wasted on validating misleading predictions but will also require more sophisticated computer architecture, e.g. an increased

use of parallel computing and stochastic training methods, which have already become standard techniques for the acceleration of deep neural network training.

- Another noticeable trend in ML is toward interpretable and explainable predictors[**?**]. Apart from the global importance of features for ML predictors, feature importance scores calculated for each input example[**?**, **?**] may help explain why a particular prediction was made for each input data point.
- Interpretable algorithms can aid in smart biocatalyst design. For instance, instead of simply screening all the possible mutations with an ML-based tool to improve a target property, researchers can make use of designing variants on the basis of the structure of a predictor using adaptive sampling[**?**].
- Such an approach favors predictors whose parameters can provide such guidance: e.g., linear predictors over more flexible yet harder to interpret artifficial neural networks (Figure 3). Linear predictors allow analytical design on the basis of the coefficients [**?**] in contrast, sophisticated predictors are usually prone to pathological behavior, i.e. sudden misclassiffication after a slight and almost imperceptible perturbation of input[**?**].
- Another promising approach is to use interpretable architectures of predictors already at the design stage, e.g. the visible neural networks[**?**]. The design of such networks is guided by the knowledge of the underlying biological mechanism, e.g. the choice of layers and the connections between layers may mimic the hierarchical organization of transcriptional regulatory factors in the cell nucleus.

51. Artículo 2: A general model to predict small molecule substrates of enzymes based on machine and deep learning

51.1. About the existence techniques

Nota 11.

- *Las enzimas han evolucionado de manera eficiente para catalizar eficientemente una o más reacciones químicas incrementando las tasas de reacción hasta más de un millón. Además la mayoría de las enzimas son promiscuas, es decir, se catalizan más, fisiológicamente irrelevantes o reacciones inofensivas [**?**, **?**]*
- *Un mapeo exhaustivo de las relaciones enzima-sustrato desempeña un papel crucial en la investigación farmacéutica y bioingeniería, por ejemplo, para la producción de medicamentos, productos químicos, alimentos y biocombustibles [**?**, **?**, **?**].*
- *La base de datos UniProt contiene entradas de más de 36 millones de enzimas diferentes, más del 99 % de ellas carecen de anotaciones de alta calidad de las reacciones catalizadas. Se están realizando esfuerzos para desarrollar métodos de alto rendicmiento para la determinación experimentarl de relaciones entre enzimas y sustratos, esfuerzos que se encuentran en su etapa inicial[**?**, **?**, **?**].*

- *Se han realizado esfuerzos por desarrollar metodos de alto rendimiento para determinar las relaciones experimentales de las relaciones enzima-sustratos. El objetivo en este trabajo es desarrollar un modelo de Machine Learning (ML) con la capacidad de predecir las relaciones enzima-sustratos a traves de todas las proteínas, por tanto contar con una herramienta que ayude a focalizar esfuerzos experimentales en pares de moléculas de pequeñas enzimas parece ser biológicamente relevantes.*

- *Los principales retos son que la representación numérica de caeda enzima que máximamente informativa para la predicción rio abajo, para ser lo más ampliamente posible, estas representaciones deben basarse de manera única en la secuencia primaria de las enzimas sin información adicional. Otro reto es que las bases de datos públicas de enzimas, solamente enlista instancias positivas, es decir, moleculas con las cuales las enzimas despliegan actividades medibes [?].*

51.2. About the ML techniques

Nota 12.

- *Para entrenar un modelo de predicción, es necesario idear una estrategia automatizada para obtener instancias adecuadas de enzimas y pequeñas moléculas que sean negativas y no se unan. Los enfoques de aprendizaje automático existentes para predecir pares enzima-sustrato fueron desarrollados específicamente para familias de enzimas pequeñas para las cuales existen conjuntos de datos de entrenamiento inusualmente completos [?]-[?].*

- *Mou et al.[?] desarrollaron modelos para predecir los sustratos de las nitrilasas bacterianas, utilizando características de entrada basadas en las estructuras tridimensionales y sitios activos de las enzimas. Entrenaron varios modelos de aprendizaje automático basados en evidencia experimental para todas las posibles combinaciones de enzima y molécula pequeña dentro del alcance de predicción de los modelos (N = 240).*

- *Yang et al.[?] predijeron el alcance de sustratos de las glicosiltransferasas de plantas entre un conjunto predefinido de moléculas pequeñas. Entrenaron un modelo basado en árboles de decisión con un conjunto de datos que cubría casi todas las posibles combinaciones de enzimas y moléculas pequeñas relevantes.*

- *Pertusi et al.[?] entrenaron cuatro máquinas de vectores de soporte (SVM) diferentes, cada una para una enzima específica. Como características de entrada, sus modelos solo utilizan información sobre los sustratos (potenciales), así como no sustratos extraídos manualmente de la literatura; no se utilizó información explícita sobre las enzimas.*

- *Roettig et al.[?] y Chevrette et al.[?] predijeron los alcances de sustratos de familias de enzimas pequeñas, entrenando modelos de aprendizaje automático con información estructural relacionada con los sitios activos de las enzimas.*

- *Visani et al.[?] implementaron un modelo de aprendizaje automático general para predecir clases EC adecuadas para un sustrato dado. Para entrenar este modelo, se utilizaron todas las clases EC que no están asociadas con un sustrato específico como puntos de*

datos negativos, lo que resultó en una baja proporción promedio de positivos a negativos de 0.0032. Visani et al. no utilizaron información sobre las enzimas más allá de la clase EC como entrada del modelo, y por lo tanto, el modelo no puede distinguir entre diferentes enzimas asignadas a la misma clase EC.

51.3. About the approuch of existing models

Nota 13.

- *Todos estos modelos no pueden aplicarse a enzimas individuales o intentan predecir sustratos solo para una enzima o familia de enzimas.*
- *Aquellos modelos que hacen predicciones para enzimas específicas se basan en datos de entrenamiento experimentales muy densos, es decir, resultados experimentales para todas o casi todas las posibles combinaciones de enzima-sustrato.*
- *Para la gran mayoría de familias de enzimas, no está disponible un conjunto de datos de entrenamiento tan extenso.*
- *Hasta el momento, no ha habido intentos publicados de formular y entrenar un modelo general que pueda aplicarse para predecir sustratos para enzimas específicas en familias de enzimas ampliamente diferentes.*
- *Los modelos de aprendizaje profundo se han utilizado para predecir funciones de enzimas ya sea prediciendo su asignación a clases EC20–22, o prediciendo dominios funcionales dentro de la secuencia de proteínas[?]*
- *Predecir directamente sustratos específicos para enzimas representa un paso importante más allá de esos métodos previos y puede ayudar a predecir la función de la enzima de manera más específica y precisa.*
- *Los enfoques de vanguardia en este dominio son basados en características, es decir, se utilizan representaciones numéricas de la proteína y la molécula del sustrato como entrada para modelos de aprendizaje automático[?]-[?].*
- *Como descripciones numéricas de la molécula del sustrato, estos enfoques utilizan representaciones SMILES30, huellas digitales expertas[?] o huellas digitales creadas con redes neuronales gráficas[?, ?].*
- *Las proteínas suelen codificarse numéricamente mediante representaciones basadas en el aprendizaje profundo de las secuencias de aminoácidos[?]-[?].*

51.4. About the work in this article

Nota 14.

- *In this work, we go beyond the current state-of-the-art by creating maximally informative protein representations, using a customized, task-specific version of the ESM-1b transformer model [?].*
- *The model contains an extra 1280-dimensional token, which was trained end-to-end to store enzyme-related information salient to the downstream prediction task.*

- *This general approach was first introduced for natural language processing37, but has not yet been applied to protein feature prediction.*
- *We created negative training examples using data augmentation, by randomly sampling small molecules similar to the substrates in experimentally confirmed enzyme-substrate pairs.*
- *we sampled all negative data points from a limited set of metabolites, the set of 1400 substrates that occur among all experimentally confirmed enzyme-substrate pairs of our dataset.*
- *Thus, we do not sample from the space of all possible alternative reactants similar to the true substrates, but only consider small molecules likely to occur in at least some biological cells.*
- *While many enzymes are rather promiscuous 2-4, it is likely that most of the potential secondary substrates are not contained in this restricted set for any given enzyme, and hence the chance of sampling false negative data points was likely small.*
- *We numerically represented all small molecules with task-specific fingerprints that we created with graph neural networks (GNNs)[**?**, **?**, **?**].*
- *A gradient-boosted decision tree model was trained on the combined protein and small molecule representations for a high-quality dataset with 18,000 very diverse, experimentally confirmed positive enzyme-substrate pairs.*
- *The resulting Enzyme Substrate Prediction model-ESP-achieves high prediction accuracy for those 1400 substrates that have been part of our training set and outperforms previously published enzyme family-specific prediction models.*

51.5. About the obtained results

Nota 15.
- *A data set with experimentally confirmed enzyme-substrate pairs using the GO annotation database for Uniprot IDs[**?**] was created.*
- *For training the machine learning models, they extracted 18,351 enzyme-substrate pairs with experimental evidence for binding, comprised of 12,156 unique enzymes and 1379 unique metabolites.*
- *Also extracted 274,030 enzyme-substrate pairs with phylogenetically inferred evidence, i.e., these enzymes are evolutionarily closely related to enzymes associated with the same reactions. These guilt by associations assignments are much less reliable than direct experimental evidence, and we only used them during pre-training to create task-specific enzyme representations-numerical vectors aimed at capturing information relevant to the prediction task from the enzyme amino acid sequences.*
- *The validations demonstrate that using phylogenetically inferred functions for the construction of appropriate enzyme representations has a positive effect on the prediction of experimentally confirmed enzyme-substrate pairs.*

- *There is no systematic information on negative enzyme-small molecule pairs, i.e., pairs where the molecule is not a substrate of the enzyme.*
- *It was hypothesized that such negative data points could be created artificially through random sampling, which is a common strategy in classification tasks that lack negative training data [**?**].*
- *Only were considered small molecules included among the experimentally confirmed enzyme-substrate pairs in our dataset.*
- *Among such a limited and biased subset, enzymes are quite specific catalyst, and therefore most of the potential secondary substrates are not included for the majority of enzymes.*
- *It was assumed that the frequency of incorrectly created negative labels is sufficiently low o not adversely affect model performance. This assumption was confirmed by the high model accuracy on independent test data.*
- *To select putatively non-binding small molecules that are structurally similar to the known substrates, it was used a similarity score based on molecular fingerprints with values ranging from 0 (no similarity) to 1 (identity; see Methods, sampling negative data points).*
- *For every positive enzyme-substrate pair, it was sampled three molecules with similarity scores between 0.75 and 0.95 to the actual substrate of the enzyme, and used them to construct negative enzyme-molecule pairs.*
- *it was opted for creating more negative data points than positive data points, as this not only provided us with more data, but it also more closely reflects the true distribution of positive and negative data points compared to a balanced distribution.*
- *Our final dataset comprises 69,365 entries. We split this data into a training set (*80 %*) and a test set (*20 %*).*
- *In many machine learning domains, it is standard practice to split the data into training and test set completely at random. However, when dealing with protein sequences, this strategy often leads to test sets with amino acid sequences that are almost identical to those of proteins in the training set.*
- *It is common practice to split such datasets into training, validation, and test sets based on protein sequences similarities [**?**].*
- *It was made sure that no enzyme in the test set has a sequence identity higher than* 80 % *compared to any enzyme in the training set. To show that despite this sequence-based partitioning, enzymes from the training and test sets follow the same distribution, we used dimensionality reduction to map all enzymes to a two-dimensional subspace and plotted the corresponding data points.*

- *To evaluate how well the final model performs for different levels of enzyme similarities, it was divided the test set further into three subsets with maximal sequence identities between* 0–40 %, 40–60 %, *and* 60–80 % *compared to all enzymes in the training set.*

- *For the numerical encoding, one classiffies bond types and calculates feature vectors with information about every atom (types, masses, valences, atomic numbers, atom charges, and number of attached hydrogen atoms)[?]. Afterwards, these identifiers are updated for a fixed number of steps by iteratively applying predefined functions to summarize aspects of neighboring atoms and bonds.*

- *After the iteration process, all identiffiers are converted into a single binary vector with structural information about the molecule. The number of iterations and the dimension of the fingerprint can be chosen freely, in this case they were set to the default values of 3 and 1024, respectively, also was created 512- and 2048-dimensional ECFPs, but these led to slightly inferior predictions. Using ECFPs can lead to identical representations for structurally very similar molecules, e.g., for some molecules that differ only by the length of a chain of carbon atoms. In our dataset, 182 out of 1379 different molecules shared an identical fingerprint with a structurally similar molecule.*

- *As an alternative to expert-crafted fingerprints such as ECFPs, neural networks can be used to learn how to map graph representations of small molecules to numerical vectors, such networks are referred to as graph neural networks (GNNs)[?, ?, ?]*

- *We trained a GNN for the binary task of predicting if a small molecule is a substrate for a given enzyme. While training for this task, the GNN is challenged to store all information about the small molecule that is relevant for solving the prediction task in a single numerical vector.*

- *After training, were extracted these 100-dimensional task-specific vectors for all small molecules in our dataset. It has been observed that pre-training GNNs for a related task can signifficantly improve model performance[?, ?].*

- *Thus, first were pre-trained the GNN for the related task of predicting the Michaelis constants KM of enzyme-substrate pairs (see Methods, pre-training indeed improved prediction performance significantly. In contrast to ECFPs, GNN-generated fingerprints lead to much fewer cases of identical representations for different molecules.*

- *In the dataset, identical fingerprints occurred for 42 out of 1379 molecules training of $ESM-1b$, 15 % of the amino acids in a protein's sequence are randomly masked and the model is trained to predict the identity of the masked amino acids. This training procedure forces the model to store both local and global information about the protein sequence in one 1280-dimensional representation vector for each individual amino acid.*

- *In order to create a single fixed-length numerical representation of the whole protein, one typically calculates the element-wise mean across all amino acid representations[?, ?, ?]. it was refered to these protein representations as ESM-1b vectors*

Nota 16. *Simply taking the element-wise mean results in information loss and does not consider the task for which the representations shall be used, which can lead to subpar performance. To overcome these issues, we created task-specific enzyme representations optimized for the prediction of enzyme-substrate pairs.*

We slightly modified the architecture of the ESM-1b model, adding one additional 1280-dimensional token to represent the complete enzyme, intended to capture information salient to the downstream prediction task. The extra token is not adding input information to the model, but it allows an easier extraction of enzyme information from the trained model. This whole-enzyme representation was updated in the same way as the regular ESM-1b amino acid representations.

- *After a predefined number of update steps, the enzyme representation was concatenated with the small molecule ECFP-vector. The combined vector was used as the input for a fully connected neural network (FCNN), which was then trained end-to-end to predict whether the small molecule is a substrate for the enzyme.*
- *This approach facilitates the construction of a single, optimized, task-specific representation. The ESM-1b model contains many parameters and thus requires substantial training data. Therefore, in the pre-training that produces the task-specific enzyme representations, we added phylogenetically inferred evidence to our training set; this resulted in a total of 287,000 data points used for training the task-specific enzyme representation.*
- *After training, we used the network to extract the 1280-dimensional task-specific representations for all enzymes in our dataset, these representations are called ESM-1bts vectors.*
- *The ESM-1b model is a state-of-the-art transformer network[**?**], trained with 27 million proteins from the UniRef dataset[**?**] in a self-supervised fashion[**?**]. This model takes an amino acid sequence as its input and puts out a numerical representation of the sequence; these representations are often referred to as protein embeddings.*
- *we estimated prediction quality on our test set when using machine learning models with each of the four combinations of enzyme and small molecule representations. In each case, we concatenated one of the two 1280-dimensional enzyme representations with one of the two small molecule representations to create a single input vector for every enzyme- small molecule pair.*
- *We used these inputs to train gradient-boosted decision tree models[**?**] for the binary classification task of predicting whether the small molecule is a substrate for the enzyme.*
- *We performed hyperparameter optimizations for all four models, including the parameters learning rate, depth of trees, number of iterations, and regularization coefficients.*
- *For this, we performed a random grid search with a 5-fold cross-validation (CV) on the training set. To challenge the model to learn to predict the substrate scope of enzymes not included in the training data, we made sure that each enzyme occurred in only one of the five subsets used for cross-validation.*

- *To account for the higher number of negative compared to positive training data, we also included a weight parameter that lowered the influence of the negative data points.*
- *After hyperparameter optimization, the models were trained with the best set of hyperparameters on the whole training set and were validated on our independent test set, which had not been used for model training or hyperparameter selection. It is noteworthy that for some input combinations, the accuracies on the test set are higher than the accuracies achieved during cross-validation.*
- *This improved performance on the test set may result from the fact that before validation on the test set, models are trained with approximately 11,000 more samples than before each cross-validation; the number of training samples has a substantial influence on model performance.*
- *To compare the gradient boosting model to alternative machine learning models, we also trained a logistic regression model and a random forest model for the task of predicting enzyme-substrate pairs from the combined ESM-1bts and GNN vectors. However, these models performed worse compared to the gradient boosting model.*
- *To investigate the importance of the added token for the observed superior performance, we alternatively re-trained the ESM-1b without such an extra token.*
- *Good predictions even for unseen enzymes It appears likely that prediction quality is best for enzymes that are highly similar to enzymes in the training set, and decreases for enzymes that are increasingly dissimilar to the enzymes used for training.*
- *How strong is that dependence? To answer this question, we first calculated the maximal enzyme sequence identity compared to the enzymes in the training set for all 2291 enzymes in the test set. Next, we split the test set into three subgroups: data points with enzymes with a maximal sequence identity to training data between 0 and* 40 %*, between* 40 % *and* 60 % *and between* 60 % *and* 80 %*.*
- *It was shown model performance is highest for enzymes that are similar to proteins in the training set. Similarly, it appears likely that the model performs better when making predictions for small molecules that are also in the training set. To test this hypothesis, we divided the test set into data points with small molecules that occurred in the training set (N = 13,459) and those with small molecules that did not occur in the training set (N = 530).*
- *The ESP model does not perform well for data points with small molecules not present in the training set. When considering only enzyme-small molecules pairs with small molecules not represented in the training set and an enzyme sequence identity level of* 0–40 % *compared to the training data, ESP achieves an accuracy of* 71 %*, ROC-AUC score 0.59, and MCC 0.15. At an enzyme sequence identity level of* 40–60 %*, accuracy improves to* 83 %*, with ROC-AUC score 0.78, and MCC 0.25 for unseen small molecules.*

- *For those test data points with small molecules not present in the training set, we wondered if a high similarity of the small molecule compared to at least one substrate in the training set leads to improved predictions, analogous to what we observed for enzymes with higher sequence identities.*
- *For each small molecules not present in the training set, we calculated the maximal pairwise similarity score compared to all substrates in the training set.*
- *we conclude that ESP only achieves high accuracies for new enzyme-small molecule pairs if the small molecule was present among the 1 400 substrates of our training set.*
- *How many training data points with identical substrates are needed to achieve good model performance? For every small molecule in the test set, we counted how many times the same molecule occurs as an experimentally confirmed substrate in the training set.*
- *Model performance increases with increased training set size The previous subsections suggest that a bigger training set with a more diverse set of enzymes and small molecules should lead to improved performance. However, using more data does not guarantee an improved model performance.*
- *To test how our model performs with different amounts of training data and to analyze if more data is expected to lead to higher generalizability, we trained the gradient boosting model with different training set sizes, ranging from* 30 % *to* 100 % *of the available training data.*
- *Internally, our trained classification model does not simply output the positive or negative class as a prediction. Mou et al.[**?**] trained four different machine learning models (logistic regression, random forest, gradient-boosted decision trees, and support vector machines) to predict substrates of bacterial nitrilases.*
- *For model training and validation, they used a dataset with all possible combinations of 12 enzymes and 20 small molecules (N = 240), randomly split into* 80 % *training data and* 20 % *test data. Instead, it outputs a prediction score between 0 and 1,Yang et al.15 published a decision tree-based model, GT-Predict, for predicting the substrate scope of glycosyltransferases of plants.All software was coded in Python57. We implemented and trained the neural networks using the deep learning library PyTorch58. We fitted the gradient boosting models using the library XGBoost50.*

Nota 17. *Deep learning-based kcat prediction enables improved enzyme-constrained model reconstruction*

- *Deep learning ha sido aplicado y mostrado un gran desempeño in el modelado de espacios químicos, expresión genética, parámetros relacionados a enzimas tales como afinidad enzimatica y números EC*
- *developed a deep learning approach (DLKcat) that uses substrate structures and protein sequences as inputs, and demonstrated its capability for the large-scale prediction of kcat val- ues for various organisms*

- *The deep learning approach DLKcat was developed by combin- ing a graph neural network (GNN) for substrates and a convolutional neural network (CNN) for proteins*

52. Articulo:Advances in Machine Learning for Directed Evolution

En este documento, se presentan algunas respuestas a preguntas y fragmentos de texto previamente proporcionados.

La evolución dirigida, que implica rondas iterativas de diversificación genética y cribado o selección fenotípica, ha surgido como una herramienta especialmente poderosa para moldear las propiedades de las proteínas en el laboratorio. La actividad específica, la estabilidad, el alcance de sustratos y la estereoselectividad de las enzimas se pueden optimizar utilizando esta técnica.

Las enzimas ofrecen soluciones a los problemas químicos más desafiantes de la vida. La capacidad de las enzimas para catalizar reacciones químicas de manera eficiente y selectiva las hace útiles no solo para los organismos anfitriones, sino también para una multitud de aplicaciones que los humanos han ideado. Como catalizadores verdes, económicos y eficientes, las enzimas han sido adoptadas por industrias que van desde la farmacéutica hasta productos de consumo, materiales, alimentos y combustibles, y se espera que su importancia siga creciendo [1–3].

La secuencia de una proteína codifica su función (*aptitud*), y la relación entre ambas se conceptualiza a menudo como una superficie en un espacio de alta dimensionalidad llamado el paisaje de aptitud de la proteína [6,7]. Nuevas proteínas se desarrollan buscando en este paisaje, comúnmente a través de un proceso de evolución dirigida [7]. La evolución dirigida avanza sometiendo una proteína que posee al menos una pequeña cantidad de la función deseada a rondas iterativas de mutagénesis y cribado, utilizando la mejor variante en cada ronda como punto de partida para la siguiente hasta que se logre el objetivo funcional (Figura 1A). A pesar de su éxito, la evolución dirigida depende de una extensa caracterización en el laboratorio, lo que constituye un cuello de botella para el desarrollo de muchas proteínas diseñadas, donde el cribado de más de unas pocas cientos o miles de variantes puede ser altamente intensivo en recursos.

Cuando se aplica a la evolución dirigida, el aprendizaje automático (ML, por sus siglas en inglés) hasta ahora se ha planteado en gran medida como un problema supervisado; es decir, dado un conjunto de secuencias de proteínas con etiquetas asociadas (por ejemplo, actividad catalítica, estabilidad, etc.), la tarea es aprender una función que pueda predecir la etiqueta de secuencias previamente no vistas (Figura 1B). Utilizando esta función, se pueden evaluar computacionalmente grandes cantidades de proteínas durante cada ciclo de evolución, lo que permite explorar mucho más a fondo el paisaje de aptitud de la proteína de lo que se podría lograr solo con cribado en el laboratorio.

Nos centramos en las formas en que los investigadores están aprovechando estrategias de aprendizaje no supervisado, es decir, estrategias que aprenden a partir de secuencias de proteínas no etiquetadas, para superar los desafíos relacionados con la recopilación de grandes conjuntos de datos de secuencias de proteínas y su función.

Comenzamos discutiendo contribuciones destacadas en el uso de secuencias de proteínas para reducir o eliminar la cantidad de datos de entrenamiento etiquetados necesarios en el aprendizaje automático supervisado. Luego destacamos trabajos que demuestran cómo los modelos entrenados solo con datos no etiquetados pueden utilizarse para generar nueva diversidad de secuencias con propiedades deseadas, así como para navegar en paisajes de aptitud de proteínas extremadamente grandes. Nuestro objetivo es hacer que esto sea accesible para la audiencia de la ingeniería de proteínas y, por lo tanto, evitamos una explicación extensa de las arquitecturas de los modelos, algoritmos y estrategias de aprendizaje que sustentan los ejemplos presentados.

Una estrategia de optimización de larga data para guiar la recopilación de datos costosos es el aprendizaje activo.

En este enfoque, un investigador entrena iterativamente un modelo con una pequeña cantidad de datos etiquados, y luego utiliza ese modelo para identificar nuevos puntos de datos para recopilar, que serían informativos y mejorarían el rendimiento del modelo. Los procesos gaussianos, que modelan su propia incertidumbre, se encuentran entre los modelos más populares para este enfoque y se han utilizado, por ejemplo, en la evolución dirigida de citocromos P450 más termoestables y variantes de channelrhodopsin para aplicaciones de optogenética.

El preentrenamiento no supervisado se basa en la suposición de que cada proteína secuenciada sigue un conjunto de reglas biofísicas y evolutivas que permiten que esa proteína sea producida y lleve a cabo una función biológica. Al entrenar modelos, que a menudo se adaptan de procesamiento de lenguaje natural (NLP) [22], en secuencias de proteínas no etiquetadas, se pueden aprender las restricciones de secuencia que resultan de estas reglas.

Los investigadores se han centrado en aumentar pequeños conjuntos de datos etiquetados con información extraída de grandes conjuntos de datos no etiquetados, una estrategia generalmente conocida como aprendizaje semi-supervisado. Cuando se aplica al campo de la ingeniería de proteínas, el aprendizaje semi-supervisado consta de una fase de aprendizaje no supervisado, a menudo denominada "preentrenamiento no supervisado.º "preentrenamiento auto-supervisado"debido a los procedimientos de entrenamiento de modelos específicos generalmente empleados, seguida de una fase de aprendizaje supervisado.

Una codificación de proteína es una representación vectorial de una secuencia de proteína necesaria para su uso en algoritmos de aprendizaje automático. Las codificaciones más simples resultan en una representación dispersa del espacio de secuencias, lo que proporciona información limitada sobre las relaciones entre secuencias y, por lo tanto, dificulta el aprendizaje [8,12]. Los embeddings de proteínas obtenidos de modelos no supervisados capturan la información aprendida durante el preentrenamiento y definen las relaciones entre las proteínas en el contexto de las restricciones de secuencia aprendidas: las secuencias similares se encontrarán más cerca unas de otras en el espacio de embeddings y, por ejemplo, se pueden inferir propiedades similares por un modelo supervisado posterior. De esta manera, los embeddings de proteínas aprendidos permiten que la información contenida en secuencias no etiquetadas se transfiera a una tarea supervisada posterior (Figura 2C-D), en principio reduciendo la cantidad de datos etiquetados necesarios en comparación con estrategias de codificación menos informativas.

Aún queda mucho por explorar en el aprendizaje semi-supervisado en la ingeniería de proteínas. Por ejemplo, hasta ahora, las arquitecturas de modelos no supervisados utilizadas

para el preentrenamiento se han adaptado principalmente de NLP. Si bien existen pruebas que sugieren que modelos de NLP más grandes entrenados en secuencias más diversas pueden mejorar los resultados de la ingeniería [23,30], también hay evidencia de que modelos mucho más pequeños con objetivos de aprendizaje más específicos para proteínas pueden lograr un rendimiento predictivo competitivo en tareas supervisadas posteriores [38]. Tampoco siempre está claro cuándo las estrategias semi-supervisadas serán superiores a las completamente supervisadas.

En medio de la creciente preocupación en la comunidad de procesamiento de lenguaje natural (NLP) acerca de los costos monetarios y energéticos de entrenar modelos de lenguaje grandes [40], el desarrollo adicional de modelos no supervisados más pequeños y la identificación de situaciones en las que el aprendizaje semi-supervisado es beneficioso son áreas importantes para investigaciones futuras.

Finalmente, también es importante señalar que, dada la inmensa magnitud del espacio de proteínas posible, el aprendizaje automático para la evolución dirigida siempre se realizará en un entorno de N relativamente bajo y nunca podrá enumerar por completo el espacio de proteínas posibles, siendo necesaria cierta cantidad de iteración. Con esta consideración, la pregunta de cómo combinar enfoques de preentrenamiento no supervisado con el aprendizaje activo se vuelve importante. Una estrategia recientemente descrita por Hie et al., que combina procesos gaussianos con embeddings de proteínas aprendidos, es un enfoque posible, al igual que una serie de algoritmos incipientes para la optimización en espacios combinatorios grandes [41–50]. En resumen, distinguir las mejores arquitecturas de modelos no supervisados y estrategias de iteración requerirá una extensa evaluación comparativa en conjuntos de datos recopilados para diferentes tareas de ingeniería de proteínas, como los proporcionados por Rao et al. [35].

Dado que las mutaciones con frecuencia llevan a la pérdida de la función, la capacidad de evitar de antemano variantes no funcionales ahorraría recursos de cribado y mejoraría significativamente la eficiencia de la evolución dirigida. Entre las aplicaciones más interesantes del aprendizaje no supervisado se encuentra la predicción "zero-shot", donde se utilizan modelos completamente no supervisados para predecir si una proteína funciona sin necesidad de un entrenamiento supervisado adicional con datos etiquetados [32,51,52]. Normalmente, esto se logra mediante el uso de un modelo generativo, que es un modelo entrenado con datos de secuencias de proteínas no etiquetados y que aprende una representación de la distribución de secuencias de proteínas permitidas (Figura 3A). Estos modelos se utilizan para consultar la probabilidad de que una nueva secuencia de proteína haya sido generada a partir de la distribución aprendida de secuencias subyacentes (Figura 3B). Si esta secuencia tiene una alta probabilidad de pertenecer a la distribución aprendida, entonces es más probable que sea una proteína funcional, y viceversa. En muchos aspectos, este enfoque es similar a la estrategia de puntuación de mutantes de proteínas basada en la conservación evolutiva, como el uso de matrices BLOSUM. Los modelos de aprendizaje automático son más capaces de capturar las interacciones epistáticas de orden superior que se cree que impregnan la evolución de las proteínas.

Mientras que entrenar un predictor de predicción "zero-shot.[en] alineamientos múltiples (MSAs) permite que un modelo generativo aprenda una representación rica de secuencias estrechamente relacionadas con un objetivo de ingeniería, si hay pocas secuencias homólogas al objetivo, entonces la distribución aprendida puede ser demasiado estrecha o dispersa para

ser utilizada de manera confiable para la predicción "zero-shot". De hecho, DeepSequence, utilizado por Riesselman et al., tuvo dificultades cuando se aplicó a proteínas para las cuales se encontraron pocas secuencias homólogas. Debido a que los modelos entrenados en bases de datos globales pueden aprender una representación más general de secuencias de proteínas, pueden ser más efectivos en tales casos. Madani et al., por ejemplo, demostraron que un gran modelo de procesamiento de lenguaje natural (NLP) entrenado en cientos de millones de secuencias de diversas familias se podía utilizar como un predictor "zero-shot"sin necesidad de recopilar secuencias de proteínas estrechamente relacionadas con el objetivo [32]. Por supuesto, todos estos estudios asumen que la aptitud del objetivo de un experimento de evolución dirigida se correlaciona bien con la aptitud optimizada evolutivamente, pero esto no siempre será el caso.

Desde la perspectiva de la evolución dirigida, un uso ideal de los modelos generativos sería identificar variantes mejoradas entre un gran número de posibilidades. Desafortunadamente, debido a que la distribución aprendida de secuencias no modela explícitamente en qué medida una proteína puede ser apta, sino solo una sensación de similitud con secuencias en las que el modelo fue entrenado, no hay expectativa de que una secuencia seleccionada sea mejorada en aptitud. Sin embargo, las estrategias recientemente propuestas que acoplan un modelo predictivo, que puede identificar variantes aptas pero requiere una costosa predicción de la aptitud de todos los candidatos, con un modelo generativo, que puede proponer variantes funcionales a partir de grandes grupos de candidatos, combinan las fortalezas de ambos y potencialmente permiten la optimización en vastos paisajes de aptitud de proteínas sin una caracterización computacional extensa [42,48–50,59]. Aunque los detalles varían, el enfoque de alto nivel de tales métodos es primero utilizar el modelo generativo para proponer un conjunto de secuencias que el modelo predictivo evaluará. Luego, se utilizan las secuencias con la aptitud predicha más alta para actualizar el modelo generativo (y posiblemente el modelo predictivo) con el fin de proponer variantes de mayor aptitud. Al repetir este ciclo, el modelo generativo propone proteínas cada vez más aptas, optimizando así la aptitud de las proteínas. Hasta ahora, tales estrategias son principalmente teóricas y deberán ser validadas a fondo mediante experimentación en el laboratorio, aunque hay algunos ejemplos de aplicación exitosa a la ingeniería de sistemas biológicos.

Al mover las costosas pruebas experimentales in silico, el aprendizaje automático (ML) amplía en gran medida nuestra capacidad para explorar el espacio de secuencias de proteínas. Si bien hasta ahora, el ML se ha planteado principalmente como un problema supervisado cuando se aplica a la evolución dirigida, también ha habido una expansión significativa en las estrategias de ML no supervisadas. Estas aproximaciones no supervisadas se pueden utilizar para limitar o eliminar la caracterización experimental requerida de proteínas, ayudar en la navegación del espacio de secuencias combinatorias y generar nueva diversidad de secuencias de proteínas, todo lo cual puede mejorar la eficiencia de las campañas de evolución dirigida. Sin embargo, el ML para la evolución dirigida sigue siendo un campo relativamente joven con mucho espacio para avances continuos. En particular, la disminución continua en el costo y el tiempo de síntesis y secuenciación de genes, así como el aumento de la potencia computacional, harán que la aplicación de métodos de ML en el laboratorio sea más factible y permitirán la expansión tanto de las bases de datos de secuencias como de secuencias-función. A medida que la disponibilidad de datos crezca, la colaboración continua y mejorada entre científicos de ML y ingenieros de proteínas será fundamental para desarrollar estrategias

de ML experimentalmente viables que avancen en el campo y fomenten una adopción más generalizada de la tecnología.

Resumen

Este artículo presenta una revisión exhaustiva de las aplicaciones del aprendizaje automático (Machine Learning, ML) dentro del campo de la inteligencia artificial (IA). Se examinan los principales tipos de aprendizaje: supervisado, no supervisado, por refuerzo y sistemas de recomendación, destacando sus aplicaciones prácticas y proponiendo ideas futuras como el "doctor virtualz la "máquina del tiempo informativa".

53. Introducción

Un agente inteligente en IA interactúa con el entorno mediante sensores y actuadores. Su inteligencia depende de la política de control que traduce entradas en acciones. ML permite alcanzar inteligencia humana simulada sin programación explícita. Aplicaciones incluyen búsqueda web, reconocimiento de fotos, y filtros de spam. Se destaca su uso en robótica autónoma, biología computacional y Big Data.

54. Aprendizaje Automático

Se definen conceptos clave del aprendizaje automático:

- Arthur Samuel lo define como la capacidad de una computadora para aprender sin ser programada.
- Tom Mitchell propone una definición formal basada en experiencia (E), tarea (T) y medida de rendimiento (P). Si el desempeño en T, medido a trav¬'es de P, mejora con la experiencia E, entonces el programa es llamando un programa de Machine Learning.

Se destaca el ejemplo del programa de damas de Samuel, que mejora jugando contra sí mismo.

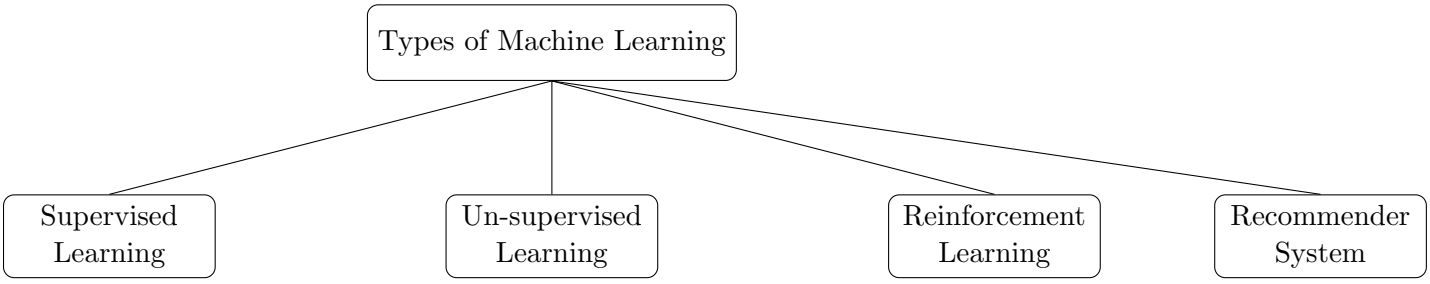


Figura 2: Types of Machine Learning

55. Tipos de Algoritmos de Aprendizaje

55.1. Aprendizaje Supervisado

Entrenamiento con datos etiquetados, comparando salida esperada con salida computada. Ejemplo: estimación del precio de viviendas.

55.2. Aprendizaje No Supervisado

Unsupervised learning is termed as learned by its own by discovering and adopting, based on the input pattern. In this learning the data are divided into different clusters and hence the learning is called a clustering algorithm Descubre patrones ocultos sin datos etiquetados. Agrupa datos en clústeres, como en Google News.

55.3. Aprendizaje por Refuerzo

Aprende mediante recompensas por buenas acciones y penalizaciones por errores, sin ejemplos explícitos. Se otorga una recompensa por una salida correcta y una penalización por una salida incorrecta. El aprendizaje por refuerzo se diferencia del aprendizaje supervisado en que nunca se presentan pares de entrada/salida correctos, ni se corrigen explícitamente las acciones subóptimas

55.4. Sistemas de Recomendación

Personalizan contenido para usuarios mediante recomendaciones basadas en contenido o colaborativas. Usado en sitios de comercio electrónico. There are mainly two approaches: content based recommendation and collaborative recommendation, which help the user for obtaining and mining data, making intelligent and novel recommendations, ethics.

56. Aplicaciones del Aprendizaje Automático

56.1. Aprendizaje No Supervisado

- **Clasificación de ADN:** Agrupamiento de individuos por genes usando microarrays.
- **Clústeres de computadores:** Organiza centros de datos eficientemente.
- **Redes sociales:** Detecta amistades, grupos, y patrones de comunicación.
- **Segmentación de mercado:** Descubre segmentos automáticamente a partir de datos de clientes.
- **Datos astronómicos:** Analiza formación de galaxias y detecta anomalías (objetos o patrones extraños).

- **Problema del cóctel:** Separa fuentes de audio combinadas usando algoritmos no supervisados.
- **Registros médicos y biología computacional:** Mejora diagnóstico, comprensión genómica y clasificación de cáncer.
- **Detección de actividad de voz (SAD):** Identifica momentos de habla versus silencio.
- **Verificación de hablantes:** Usa análisis acústico para autenticación.

56.2. Aprendizaje Supervisado

- **Correo electrónico:** Respuestas automáticas, organización de carpetas, resumen de hilos, y filtro de spam.
- **Reconocimiento de escritura:** Identifica direcciones en sobres.
- **Reconocimiento facial y de voz:** Aplicado en seguridad y redes sociales.
- **Recuperación de información:** Búsqueda eficiente y personalizada.
- **Sistemas operativos:** Predicen apps frecuentes para carga rápida.
- **Detección de intrusos y anómalias:** Usa secuencias de acciones para detectar comportamientos anormales.
- **Clasificación de textos:** Asigna documentos a categorías temáticas.
- **Optimización de centros de datos:** Usa redes neuronales para eficiencia energética.
- **Radio cognitiva:** Mejora procesamiento de señales mediante reducción de dimensionalidad y SVM.
- **Finanzas computacionales:** Predice movimientos del mercado bursátil.
- **Interfaces cerebro-máquina (BCI):** Permite controlar dispositivos con actividad cerebral.
- **Producción musical:** Clasifica géneros, transcribe, detecta ritmo e instrumentos.

56.3. Sistemas de Recomendación

- **Aprendizaje móvil:** Ofrece contenido educativo personalizado.
- **Publicidad computacional:** Asocia usuarios con anuncios en contexto.
- **Análisis de sentimientos:** Clasifica opiniones como positivas o negativas.
- **Minería de bases de datos:** Extrae patrones de grandes volúmenes de datos.
- **Programas auto-personalizables:** Aprenden preferencias del usuario y adaptan interfaces.

56.4. Aprendizaje por Refuerzo

- **Predicción de tráfico:** Sistemas que estiman condiciones futuras de tráfico.
- **Juegos de computadora:** IA que mejora experiencia de juego.
- **Maquinaria autónoma:** Aprenden tareas como volar helicópteros.
- **Análisis bursátil:** Usa SVM y refuerzo para tomar decisiones financieras.
- **Ambientes de aprendizaje ubicuo:** Simulaciones realistas para evaluar habilidades clínicas.

57. Impresiones y Perspectivas

Se observa que la capacidad de análisis de datos masivos ha impulsado el desarrollo de agentes autónomos. Se destaca el papel del aprendizaje continuo y la necesidad de conjuntos de datos actualizados. Entre las propuestas futuras se incluyen la máquina del tiempo informativa y el doctor virtual.

58. Conclusión

El aprendizaje automático ha demostrado ser esencial para la automatización inteligente, con aplicaciones en múltiples disciplinas. Aunque hay limitaciones en la calidad y disponibilidad de los datos, el campo sigue creciendo. Se destaca la necesidad de mejorar continuamente los algoritmos y entrenarlos con datos diversos y actualizados.

59. Resumen

La regresión logística es una forma eficiente y poderosa de analizar el efecto de un grupo de variables independientes sobre un resultado binario cuantificando la contribución única de cada variable independiente, por otra parte la regresión logística identifica iterativamente la combinación lineal más fuerte de variables con la mayor probabilidad de detectar el resultado observado.

Es importante considerar en una regresión logística la *selección de variables independientes* para esto uno debe guiarse por factores tales como teoría existente, investigaciones empíricas previas, consideraciones clínicas y análisis estadísticos univariados, reconociendo las posibles variables de confusión que deben ser consideradas.

Los supuestos básicos que deben cumplirse para la regresión logística incluyen

- independencia de errores,
- linealidad en el *logit* para variables continuas,
- ausencia de multicolinealidad y

- falta de valores atípicos fuertemente influyentes. Adicionalmente,
- existencia de un número adecuado de eventos por variable independiente para evitar un modelo sobreajustado, con un mínimo comúnmente recomendado de "reglas prácticas" que van de 10 a 20 eventos por covariable.

Respecto a las estrategias de construcción de modelos, los tres tipos generales son:

- directa/estándar,
- secuencial/jerárquica y
- por pasos/estadística,

cada uno con un énfasis y propósito diferente. Antes de llegar a conclusiones definitivas a partir de los resultados de cualquiera de estos métodos, se debe cuantificar formalmente la validez interna del modelo (es decir, su replicabilidad dentro del mismo conjunto de datos) y su validez externa (es decir, su generalizabilidad más allá de la muestra actual).

El ajuste general del modelo de regresión logística a los datos de muestra se evalúa utilizando varias *medidas de bondad de ajuste*, donde un mejor ajuste se caracteriza por una menor diferencia entre los valores observados y los valores predichos por el modelo. También se recomienda el uso de *estadísticas de diagnóstico* para evaluar aún más la adecuación del modelo. Finalmente, los resultados para las variables independientes suelen reportarse como *razones de momios* (odds ratios, ORs) con *intervalos de confianza* (IC) del 95 %.

60. Importancia de la regresión en investigación

La **regresión** es un método valioso de investigación debido a su versátil aplicación en diferentes contextos de estudio. Por ejemplo, se puede utilizar para examinar asociaciones entre un resultado y varias variables independientes (también comúnmente conocidas como covariables, predictores o variables explicativas)[**?**], o para determinar qué tan bien puede predecirse un resultado a partir de un conjunto de variables independientes[**?**, **?**]. Adicionalmente, uno puede estar interesado en controlar el efecto de variables independientes específicas, particularmente aquellas que actúan como variables de confusión (es decir, cuya relación tanto con el resultado como con otra variable independiente oscurece la relación entre esa variable independiente y el resultado)[**?**, **?**]. Esta última aplicación es especialmente útil en contextos donde no es posible asignar aleatoriamente sujetos a grupos de tratamiento, como sucede en investigaciones observacionales. Con asignación aleatoria, normalmente se puede ejercer un control adecuado sobre las variables de confusión, ya que los grupos aleatorizados tienden a tener una distribución equitativa o balanceada de dichas variables[**?**].

61. Regresión Logística

Existen diferentes tipos de regresión, dependiendo de los objetivos de investigación y del formato de las variables, siendo la regresión lineal una de las más utilizadas. La *regresión*

lineal analiza resultados continuos (es decir, aquellos que pueden sumarse, restarse, multiplicarse o dividirse de manera significativa, como el peso) y asume que la relación entre el resultado y las variables independientes sigue una forma funcional determinada. Sin embargo, generalmente es más deseable determinar la influencia de múltiples factores al mismo tiempo, ya que de este modo se pueden observar las contribuciones únicas de cada variable después de controlar por los efectos de las demás. En este caso, la regresión lineal multivariada es la opción adecuada.

La ecuación básica para la regresión lineal con múltiples variables independientes es:

$$\hat{Y} = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_i X_i. \tag{6}$$

Los componentes de esta ecuación son los siguientes:

- $\hat{Y}$ es el resultado continuo estimado.
- $\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_i X_i$ es la ecuación de regresión lineal para las variables independientes del modelo, donde:
  - β_0 es la ordenada al origen o punto en el que la línea de regresión toca el eje vertical Y. Se considera un valor constante.
  - $\beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_i X_i$ es el valor de cada variable independiente (X_i) ponderado por su respectivo coeficiente beta (β). Los coeficientes beta determinan la pendiente de la línea de regresión, cuanto mayor sea el coeficiente beta, más fuerte es la contribución de dicha variable al resultado.

Para una variable binaria, como la mortalidad, la regresión logística es el método usualmente elegido, la regresión logística puede incluir una o múltiples variables independientes, aunque examinar múltiples variables es generalmente más informativo, ya que permite revelar la contribución única de cada variable ajustando por las demás. La regresión logística tiene ecuación:

$$\text{Probabilidad del resultado}(\hat{Y}_i) = \frac{e^{\beta_0+\beta_1 X_1+\beta_2 X_2+\ldots+\beta_i X_i}}{1 + e^{\beta_0+\beta_1 X_1+\beta_2 X_2+\ldots+\beta_i X_i}}. \tag{7}$$

62. Transformación logística y elección de variables independientes

Un aspecto importante de la regresión logística es que conserva muchas características de la regresión lineal en su análisis de resultados binarios. Sin embargo, existen diferencias clave entre las dos ecuaciones:

1. $\hat{Y}_i$ representa la probabilidad estimada de pertenecer a una de las dos categorías binarias del resultado (categoría i) en lugar de representar un resultado continuo estimado.

2. $e^{\beta_0+\beta_1 X_1+\beta_2 X_2+\ldots+\beta_i X_i}$ representa la ecuación de regresión lineal para las variables independientes expresadas en la escala *logit*.

La razón de esta transformación *logit* radica en los parámetros básicos del modelo de regresión logística, un resultado binario expresado como probabilidad debe estar entre 0 y 1 [**?**]. La escala logit resuelve este problema al transformar matemáticamente la ecuación de regresión lineal original para producir el logit (o logaritmo natural) de las razones de momios (odds) de estar en una categoría ($\hat{Y}$) frente a la otra categoría ($1-\hat{Y}$):

$$\ln\left(\frac{\hat{Y}}{1-\hat{Y}}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_i X_i \tag{8}$$

En el contexto de estas ecuaciones, la regresión logística identifica, mediante ciclos iterativos, la combinación lineal más fuerte de variables independientes que aumente la probabilidad de detectar el resultado observado —un proceso conocido como estimación de máxima verosimilitud [**?**, **?**]. Para asegurar que la regresión logística produzca un modelo preciso, se deben considerar factores críticos como la selección de variables independientes y la elección de la estrategia de construcción del modelo.

62.1. Variables independientes

1. **Criterios de selección.** Seleccionar cuidadosamente las variables independientes es un paso esencial. Aunque la regresión logística es bastante flexible y permite distintos tipos de variables (continuas, como la edad; ordinales, como escalas de dolor analógico visual; y categóricas, como la raza), siempre debe justificarse la selección de variables utilizando teoría bien establecida, investigaciones previas, observaciones clínicas, análisis estadístico preliminar, o una combinación razonada de estas opciones.

   Por ejemplo, se podría comenzar con un gran número de variables independientes potenciales con base en estudios previos y experiencia clínica en el departamento de urgencias, y luego analizar diferencias entre grupos mediante estadística univariada con un nivel de error tipo I más relajado (por ejemplo, $p \leq 0{,}25$) para determinar qué variables deben incluirse en el modelo de regresión logística. Usar un valor de p menos estricto en esta etapa protege contra la exclusión de variables potencialmente importantes. Alternativamente, uno podría optar por incluir todas las variables independientes relevantes independientemente de sus resultados univariados, ya que puede haber variables clínicamente importantes que merezcan inclusión a pesar de su desempeño estadístico. Sin embargo, siempre debe tenerse en cuenta que incluir demasiadas variables independientes en el modelo puede conducir a un resultado matemáticamente inestable, con menor capacidad de generalización más allá de la muestra actual del estudio.[1]

   Una parte clave del proceso de selección de variables es reconocer y considerar el papel de los posibles factores de confusión. Como se describió previamente, las variables de

[1] Referencias 2,3

confusión son aquellas cuya relación tanto con el resultado como con otra variable independiente oculta la verdadera asociación entre esa variable independiente y el resultado.[2]

Por ejemplo, el nivel socioeconómico (SES) podría confundir la relación entre la raza y las visitas anuales a emergencias, debido a su asociación con ambas variables (es decir, ciertos grupos raciales tienden a estar sobrerrepresentados en algunas categorías de SES y los pacientes más pobres pueden usar más frecuentemente los servicios de urgencias). No obstante, como este tipo de asociaciones causales no siempre son evidentes, se recomienda evaluarlas formalmente durante el proceso de selección de variables, a fin de garantizar que se modelen adecuadamente. Los diagramas de análisis de trayectorias pueden ser particularmente útiles en este sentido.[3]

Independientemente del método para seleccionar las variables independientes, deben cumplirse ciertos supuestos básicos al aplicar regresión logística. Un supuesto es la **independencia de los errores**, lo cual significa que todos los resultados del grupo de muestra deben ser independientes entre sí (por ejemplo, que no haya respuestas duplicadas). Si los datos incluyen mediciones repetidas u otros resultados correlacionados, los errores también estarán correlacionados y el supuesto se violará.[4] Existen otros métodos para analizar datos correlacionados mediante técnicas de regresión logística, pero van más allá del alcance de este artículo; para más información, los lectores pueden consultar a Stokes et al.,[5] Newgard et al.,[6] y Allison.[7]

Un segundo supuesto es la **linealidad en el logit** para las variables continuas independientes (por ejemplo, edad), lo que significa que debe existir una relación lineal entre estas variables y sus respectivos resultados transformados en logit. Hay diversas formas de verificar este supuesto, siendo una técnica común la creación de un término de interacción entre cada variable continua independiente y su logaritmo natural. Si alguno de estos términos es estadísticamente significativo, el supuesto se considera violado.[8]

Las soluciones incluyen codificación dicotómica de la variable independiente,[9] o su transformación estadística a otra escala.[10]

Un tercer supuesto es la **ausencia de multicolinealidad**, o redundancia entre variables independientes (por ejemplo, peso e índice de masa corporal [IMC] están correlacionados, por lo que no deben incluirse en el mismo modelo). Un modelo de regresión logística con variables independientes altamente correlacionadas usualmente genera errores estándar grandes para los coeficientes beta (o pendientes) estimados. La solución común es eliminar una o más variables redundantes.[11]

[2]Referencias 1,3
[3]Referencia 1
[4]Referencia 2
[5]Referencia 5
[6]Referencias 6,7
[7]Referencia 8
[8]Referencias 2,3
[9]Referencia 3
[10]Referencias 2,3
[11]Referencia 2

El supuesto final es la **ausencia de valores atípicos altamente influyentes**, es decir, casos en los que el resultado predicho para un miembro de la muestra difiere considerablemente de su valor real...

...resultado. Si hay demasiados valores atípicos, la precisión general del modelo puede verse comprometida. La detección de valores atípicos se realiza examinando los residuales (es decir, la diferencia entre los valores predichos y los resultados reales) junto con estadísticas diagnósticas y gráficas.[12] Luego, se puede comparar el ajuste general del modelo y los coeficientes beta estimados con y sin los casos atípicos. Dependiendo de la magnitud del cambio, uno podría conservar los valores atípicos cuyo efecto no sea dramático[13] o eliminar aquellos con una influencia particularmente fuerte sobre el modelo.[14]

Además de comprobar que se cumplan los supuestos anteriores, se puede considerar incluir términos de interacción que combinen dos o más variables independientes. Por ejemplo, es posible que la interacción entre la edad y la raza de los pacientes sea más importante para explicar un resultado que cualquiera de estas variables por separado[15] (por ejemplo, la relación entre la edad y la mortalidad relacionada con trauma varía entre asiáticos, blancos e hispanos). Sin embargo, los términos de interacción pueden complicar innecesariamente el modelo de regresión logística sin aportar mucho beneficio.[16] Por ello, se debe pensar cuidadosamente antes de incluirlos, obteniendo orientación de diagnósticos estadísticos (por ejemplo, observando cuánto cambian los coeficientes beta estimados, o pendientes, de una variable independiente al añadir otra al modelo), y evaluando si las interacciones tienen sentido clínico.[17]

2. Número de variables a incluir. Como parte del proceso de selección de qué variables independientes incluir, también se debe decidir cuántas. El reto es seleccionar el menor número posible de variables independientes que expliquen mejor el resultado sin descuidar las limitaciones del tamaño de muestra.[18] Por ejemplo, si se seleccionan 50 personas para el estudio y se incluyen 50 variables independientes en el análisis de regresión logística, el resultado es un modelo sobreajustado (y por tanto inestable). En términos generales, un modelo sobreajustado tiene coeficientes beta estimados para las variables independientes mucho mayores de lo que deberían ser, además de errores estándar más altos de lo esperado.[19] Este tipo de situación genera inestabilidad en el modelo porque la regresión logística requiere más resultados que variables independientes para poder iterar soluciones diferentes en busca del mejor ajuste a través del método de máxima verosimilitud.[20]

[12]Referencias 2,3
[13]Referencia 3
[14]Referencias 2,3
[15]Referencia 3
[16]Referencias 2,3
[17]Referencia 3
[18]Referencias 2,3
[19]Referencia 3
[20]Referencias 2,3

Entonces, ¿cuál es el número correcto de resultados para evitar un modelo sobreajustado? Aunque no existe un estándar universalmente aceptado, hay algunas "reglas generales" derivadas en parte de estudios de simulación. Una de estas reglas sugiere que por cada variable independiente, debe haber al menos 10 resultados por cada categoría binaria (por ejemplo, vivo/muerto), siendo el resultado menos frecuente el que determina el número máximo de variables independientes.[21] Por ejemplo, en un estudio de mortalidad por sepsis, si se asume que 30 pacientes murieron y 50 sobrevivieron, el modelo podría acomodar, como máximo, tres variables independientes (ya que 30 es el resultado menos frecuente). Algunos estadísticos recomiendan una "regla general" aún más estricta de 20 resultados por variable independiente, dado que una relación más alta tiende a mejorar la validez del modelo.[22] Sin embargo, el tema no está completamente resuelto y algunos argumentan que menos de 10 resultados por variable pueden ser apropiados en ciertos contextos de investigación.[23]

Estrategias de Construcción del Modelo

Además de la cuidadosa selección de las variables independientes, se debe elegir el tipo adecuado de modelo de regresión logística para el estudio. De hecho, seleccionar una estrategia de construcción del modelo está estrechamente relacionado con la elección de variables independientes, por lo que estos dos componentes deben considerarse simultáneamente al planear un análisis de regresión logística.

Existen tres enfoques generales para la construcción del modelo que se aplican a las técnicas de regresión en general, cada uno con un énfasis y propósito diferente: directo (es decir, completo, estándar o simultáneo), secuencial (es decir, jerárquico) y paso a paso (es decir, estadístico). Estas estrategias de construcción no son necesariamente intercambiables, ya que pueden producir diferentes medidas de ajuste del modelo y diferentes estimaciones puntuales para las variables independientes a partir de los mismos datos. Por lo tanto, identificar el modelo apropiado para los objetivos del estudio es extremadamente importante.

El enfoque directo es una especie de valor por defecto, ya que introduce todas las variables independientes en el modelo al mismo tiempo y no hace suposiciones sobre el orden o la importancia relativa de dichas variables.[24] Por ejemplo, al analizar la mortalidad a 30 días en pacientes sépticos admitidos por el departamento de emergencias (ED), si se identifican 10 variables independientes para incluir, entonces las 10 se introducen en el modelo simultáneamente y tienen la misma importancia al inicio del análisis.

El enfoque directo es más adecuado si no existen hipótesis previas sobre cuáles variables tienen mayor relevancia que otras. De lo contrario, se puede considerar el uso de regresión secuencial/jerárquica, en la cual las variables se añaden secuencialmente para evaluar si mejoran el modelo de acuerdo a un orden predeterminado de prioridad.[25] Por ejemplo, se podría iniciar introduciendo la edad en el modelo, suponiendo que es el predictor más fuerte de mortalidad a 30 días en pacientes admitidos por sepsis, seguido de edad más comorbilidades, luego edad, comorbilidades y volumen de casos de sepsis en el ED, y así sucesivamente.

[21] Referencias 9,10
[22] Referencia 11
[23] Referencia 3
[24] Referencias 1,2
[25] Referencias 1,2

Aunque este enfoque es útil para clarificar patrones causales entre variables independientes y resultados, puede volverse complejo conforme aumentan los patrones causales, dificultando así la obtención de conclusiones definitivas sobre los datos en algunos casos.[26]

En contraste con los dos métodos anteriores, la regresión paso a paso identifica variables independientes que deben mantenerse o eliminarse del modelo con base en criterios estadísticos predefinidos que están influenciados por las características únicas de la muestra analizada.[27] Existen distintos tipos de técnicas paso a paso, incluyendo selección hacia adelante (por ejemplo, edad, comorbilidades, volumen de casos de sepsis en el ED, y otras variables independientes son introducidas una por una en el modelo para mortalidad por sepsis a 30 días, hasta que no se identifiquen más variables adicionales que contribuyan significativamente al resultado) y eliminación hacia atrás (por ejemplo, edad, comorbilidades, volumen de casos de sepsis en el ED, y otras variables se introducen todas simultáneamente en el modelo, y luego se eliminan una a una aquellas con contribuciones no significativas). con una contribución no significativa al resultado son eliminadas una por una hasta que sólo queden las variables estadísticamente significativas).[28] Otra estrategia de construcción del modelo que es conceptualmente similar a la regresión por pasos se llama "selección del mejor subconjunto", en la que se comparan modelos separados con diferentes números de variables independientes (por ejemplo, edad sola, edad más comorbilidades, comorbilidades más volumen de casos de sepsis en urgencias) para determinar el mejor ajuste según lineamientos preestablecidos.[29]

Una Nota de Precaución

Aunque la regresión por pasos se usa frecuentemente en la investigación clínica, su uso es algo controvertido porque se basa en una selección automatizada de variables que tiende a aprovechar factores aleatorios en una muestra dada.[30] Además, la regresión por pasos puede producir modelos que no parecen completamente razonables desde una perspectiva biológica.[31] Ante estas preocupaciones, algunos argumentan que la regresión por pasos se reserva mejor para el tamizaje preliminar o únicamente para pruebas de hipótesis,[32] como en casos de resultados novedosos y una comprensión limitada de las contribuciones de las variables independientes.[33] Sin embargo, otros señalan que los métodos por pasos no son en sí el problema (y de hecho pueden ser bastante efectivos en ciertos contextos); en cambio, el verdadero problema es una interpretación descuidada de los resultados sin valorar completamente los pros y contras de este enfoque. Por tanto, si uno elige crear un modelo por pasos, es importante validar posteriormente los resultados antes de sacar conclusiones. No obstante, debe destacarse que todos los tipos de modelos requieren validación formal antes de que se consideren definitivos para uso futuro, ya que se espera que los modelos funcionen

[26]Referencia 1
[27]Referencias 2,3
[28]Referencias 1,3
[29]Referencia 3
[30]Referencia 2
[31]Referencia 3
[32]Referencia 2
[33]Referencia 3

mejor con la muestra original que con muestras subsiguientes.[34]

Validación Interna y Externa del Modelo

Al validar modelos de regresión logística, existen numerosos métodos entre los cuales elegir, cada uno más o menos apropiado según los parámetros del estudio como el tamaño de muestra. Para establecer la validez interna (confirmación de resultados del modelo con el mismo conjunto de datos), los métodos comunes incluyen: 1) el método de retención, o división de la muestra en dos subgrupos antes de la construcción del modelo, con el grupo de "entrenamiento" usado para crear el modelo de regresión logística y el grupo de "prueba" usado para validarlo;[35] 2) validación cruzada k-fold o división de la muestra en k subgrupos de igual tamaño para propósitos de entrenamiento y validación;[36] 3) validación cruzada "uno fuera" (leave-one-out), una variante del método k-fold donde el número de particiones es igual al número de sujetos en la muestra;[37] y 4) diferentes formas de bootstrapping (es decir, obtener submuestras repetidas con reemplazo de toda la muestra).[38]

Además de validar internamente el modelo, uno debería intentar validarlo externamente en un nuevo entorno de estudio como una prueba adicional de su viabilidad estadística y utilidad clínica.[39] Si los resultados de la validación interna o externa presentan alguna alerta (por ejemplo, el modelo tiene bajo rendimiento para cierto subgrupo de pacientes), se recomienda hacer ajustes al modelo según sea necesario, o definir explícitamente cualquier restricción para el uso futuro del modelo.[40]

Interpretación de los Resultados del Modelo

1. Evaluación del Ajuste General del Modelo. Una vez que se ha creado el modelo de regresión logística, se determina qué tan bien se ajusta a los datos de la muestra en su totalidad. Dos de los métodos más comunes para evaluar el ajuste del modelo son la prueba de chi-cuadrado de Pearson y la desviación residual. Ambas miden la diferencia entre los resultados observados y los resultados predichos por el modelo, donde un mal ajuste del modelo se indica mediante valores de prueba elevados, lo que señala una diferencia mayor. Sin embargo, la precisión de estas medidas depende de contar con un número adecuado de observaciones para los diferentes patrones de variables independientes.[41]

Otra medida comúnmente utilizada del ajuste del modelo es la prueba de bondad de ajuste de Hosmer-Lemeshow, que divide a los sujetos en grupos iguales (a menudo de 10) según su probabilidad estimada del resultado. El decil más bajo está compuesto por aquellos que tienen menor probabilidad de experimentar el resultado. Si el modelo tiene buen ajuste, los sujetos que experimentaron el resultado principal (por ejemplo, mortalidad por sepsis a

[34]Referencia 3
[35]Referencias 12,13
[36]Referencia 13
[37]Referencia 13
[38]Referencias 13,14
[39]Referencias 12,15
[40]Referencia 15
[41]Referencias 3, 16, 17

los 30 días) caerán en su mayoría en los deciles de mayor riesgo. Un modelo con mal ajuste resultará en sujetos distribuidos de manera más uniforme a lo largo de los deciles de riesgo para ambos resultados binarios.[42]

Las ventajas de las pruebas de Hosmer-Lemeshow incluyen su aplicación sencilla y facilidad de interpretación.[43] Las limitaciones incluyen la dependencia de las pruebas sobre cómo se definen los puntos de corte de los grupos[44] y los algoritmos computacionales utilizados,[45] así como una menor capacidad para identificar modelos con mal ajuste en ciertas circunstancias.[46] Otras alternativas menos comunes para evaluar el ajuste del modelo son descritas por Hosmer et al.[47] y Kuss.[48]

Aunque los índices de ajuste del modelo son componentes esenciales de la regresión logística, también se deben usar estadísticas de diagnóstico antes de sacar conclusiones sobre la adecuación del modelo final. Estas estadísticas ayudan a determinar si el modelo permanece intacto en todas las configuraciones posibles de las variables independientes.[49] Aunque una visión detallada de los métodos de diagnóstico excede el alcance de este artículo, se puede consultar a Hosmer y Lemeshow[50] para obtener más información.

Como forma de ampliar los resultados del ajuste del modelo y de las estadísticas diagnósticas, también se puede evaluar la capacidad del modelo para discriminar entre grupos. Las formas comunes de hacer esto incluyen 1) tablas de clasificación, donde la pertenencia a un grupo dentro de una categoría binaria del resultado se predice usando probabilidades estimadas y puntos de corte predefinidos,[51] y 2) el área bajo la curva característica operativa del receptor (AUROC), donde un valor de 0.5 significa que el modelo no es mejor que el azar para discriminar entre los sujetos que tienen el resultado y los que no, y un valor de 1.0 indica que el modelo discrimina perfectamente entre sujetos. El AUROC se usa a menudo cuando se desean considerar diferentes puntos de corte para la clasificación y así maximizar tanto la sensibilidad como la especificidad.[52]

2. Interpretación de los Resultados de Variables Individuales. Dentro del contexto del modelo de regresión logística, las variables independientes usualmente se presentan como razones de momios (ORs, por sus siglas en inglés).[53] Las ORs revelan la fuerza de la contribución de la variable independiente al resultado y se definen como las probabilidades de que ocurra el resultado ($\hat{Y}$) frente a que no ocurra...
($1 - \hat{Y}$) para cada variable independiente. La relación entre la razón de momios (OR) y el coeficiente beta estimado de la variable independiente se expresa como OR = e^{β_i}. Con base en esta fórmula, un cambio de una unidad en la variable independiente multiplica la

[42]Referencias 2, 3
[43]Referencias 3, 16
[44]Referencias 10, 11
[45]Referencia 17
[46]Referencias 3, 16
[47]Referencias 16, 17
[48]Referencia 17
[49]Referencia 3
[50]Referencia 3
[51]Referencias 3, 21
[52]Referencias 3, 18
[53]Referencia 3

probabilidad del resultado por la cantidad contenida en e^{β_i}.[54]

Para un modelo de regresión logística con solo una variable independiente, la OR se considera "no ajustada" porque no hay otras variables cuya influencia deba ser ajustada o restada. Para fines ilustrativos, supongamos que el resultado es mortalidad intrahospitalaria después de una lesión traumática, y que la única variable independiente es la edad del paciente, clasificada en mayores o menores de 65 años, con la categoría más reciente como grupo de referencia (o el grupo con el que se comparan todas las demás categorías de variables independientes). Una OR de 1.5 significa que para los pacientes mayores, las probabilidades de morir son 1.5 veces mayores que para los pacientes más jóvenes (grupo de referencia). Expresado de otro modo, hay un aumento del $(1{,}5-1{,}0)\times 100\,\% = 50\,\%$ en las probabilidades de morir en el hospital después de una lesión traumática para pacientes mayores frente a los más jóvenes.

En contraste, si el modelo de regresión logística incluye múltiples variables independientes, las OR ahora son "ajustadas" porque representan la contribución única de la variable independiente después de ajustar (o restar) los efectos de las otras variables en el modelo. Por ejemplo, si el escenario de mortalidad intrahospitalaria posterior a un trauma incluye edad más sexo, IMC y comorbilidades, la OR ajustada para la edad representa su contribución única a la mortalidad intrahospitalaria cuando las otras tres variables se mantienen constantes. Como resultado, las OR ajustadas suelen ser menores que sus contrapartes no ajustadas.

Interpretar las OR también depende de si la variable independiente es continua o categórica. Para las variables continuas, primero se debe identificar una unidad de medida significativa que exprese mejor el grado de cambio en el resultado asociado con esa variable independiente.[55] Usando la ilustración anterior de mortalidad intrahospitalaria con la edad mantenida en su escala continua original y seleccionando incrementos de 10 años como la unidad de cambio, uno interpretaría los resultados de la siguiente manera: "Por cada 10 años que envejece un paciente, las probabilidades de morir en el hospital después de una lesión traumática aumentan 1.5 veces, o un 50 %".

Finalmente, los intervalos de confianza (IC) al 95 % se informan rutinariamente junto con las OR como una medida de precisión (es decir, si los hallazgos probablemente se mantendrán en la población no observada). Si el IC cruza 1.00, es posible que no haya una diferencia significativa en esa población. Por ejemplo, si la OR de 1.5 para la edad tiene un IC del 95 % de 0.85 a 2.3, no se puede afirmar de manera concluyente que la edad sea un contribuyente significativo a la mortalidad intrahospitalaria tras una lesión traumática.

62.2. Resumen

Las técnicas de regresión son versátiles en su aplicación a la investigación médica porque pueden medir asociaciones, predecir resultados y controlar los efectos de variables de confusión. Como una de estas técnicas, la regresión logística es una forma eficiente y poderosa de analizar el efecto de un grupo de variables independientes sobre un resultado binario al cuantificar la contribución única de cada variable independiente. Utilizando componentes

[54] Referencias 2,3

[55] Referencia 3

de la regresión lineal reflejados en la escala logit, la regresión logística identifica de manera iterativa la combinación lineal más fuerte de variables con la mayor probabilidad de detectar el resultado observado.

Consideraciones importantes al realizar regresión logística incluyen la selección de variables independientes, asegurarse de que se cumplan los supuestos relevantes y elegir una estrategia adecuada para la construcción del modelo. Para la selección de variables independientes, se deben considerar factores como la teoría aceptada, investigaciones empíricas previas, consideraciones clínicas y análisis estadísticos univariantes, reconociendo las posibles variables de confusión que deben ser tenidas en cuenta.

Los supuestos básicos que deben cumplirse para la regresión logística incluyen: independencia de los errores, linealidad en el logit para variables continuas, ausencia de multicolinealidad y ausencia de valores atípicos altamente influyentes. Además, debe haber un número adecuado de eventos por variable independiente para evitar un modelo sobreajustado, con una regla general recomendada que oscila entre 10 y 20 eventos por covariable.

Respecto a las estrategias de construcción del modelo, existen tres tipos generales: directa/estándar, secuencial/jerárquica y por pasos/estadística, cada una con un énfasis y propósito diferente. Antes de llegar a conclusiones definitivas a partir de los resultados de cualquiera de estos métodos, se debe cuantificar formalmente la validez interna (i.e., replicabilidad dentro del mismo conjunto de datos) y la validez externa (i.e., generalización más allá de la muestra actual).

El ajuste general del modelo de regresión logística a los datos de muestra se evalúa utilizando diversas medidas de bondad de ajuste, siendo mejor el ajuste cuanto menor sea la diferencia entre los valores observados y los valores predichos por el modelo. También se recomienda el uso de estadísticas de diagnóstico para evaluar adecuadamente el modelo. Finalmente, los resultados para las variables independientes se reportan típicamente como razones de momios (odds ratios, OR) con intervalos de confianza del 95 % (ICs).

62.3. Tipos de regresión y fundamentos de la regresión logística

Existen diferentes tipos de regresión según los objetivos de la investigación y el formato de las variables, siendo la regresión lineal una de las más utilizadas. La regresión lineal analiza resultados continuos (es decir, aquellos que pueden sumarse, restarse, multiplicarse y dividirse significativamente, como el peso) y asume que la relación entre el resultado y las variables independientes sigue una línea recta (por ejemplo, a medida que aumentan las calorías consumidas, aumenta el peso).

Para evaluar el efecto de una sola variable independiente sobre un resultado continuo (por ejemplo, el efecto del consumo de calorías sobre el aumento de peso), se realizaría una regresión lineal simple. Sin embargo, normalmente es más deseable determinar la influencia de múltiples factores al mismo tiempo (por ejemplo, calorías consumidas, días de ejercicio por semana y edad en el aumento de peso), ya que esto permite ver las contribuciones únicas de cada variable después de controlar los efectos de las demás. En este caso, la regresión lineal multivariada es la opción adecuada.

La ecuación básica para la regresión lineal con múltiples variables independientes es:

$$\hat{Y} = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_i X_i \tag{9}$$

- β_0 es la ordenada al origen, o el punto en el que la línea de regresión toca el eje vertical Y. Se considera un valor constante.

- $\beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_i X_i$ representa el valor de cada variable independiente (X_i) ponderado por su coeficiente beta (β). Estos coeficientes indican la pendiente de la línea de regresión o cuánto aumenta el resultado por cada unidad adicional en el valor de la variable independiente. Cuanto mayor es el coeficiente beta, mayor es la contribución de su variable independiente correspondiente al resultado.

A pesar de su uso común, la regresión lineal no es adecuada para ciertos tipos de resultados médicos. Para eventos binarios, como la mortalidad, la regresión logística es el método habitual de elección. Al igual que la regresión lineal, la regresión logística puede incluir una o varias variables independientes, siendo generalmente más informativa la evaluación de múltiples variables porque permite ver las contribuciones únicas de cada una tras ajustar por las otras.

La identificación de estas contribuciones en la regresión logística comienza con la siguiente ecuación:

$$\text{Probabilidad del resultado } (\hat{Y}_i) = \frac{e^{\beta_0+\beta_1 X_1+\beta_2 X_2+\ldots+\beta_i X_i}}{1 + e^{\beta_0+\beta_1 X_1+\beta_2 X_2+\ldots+\beta_i X_i}} \tag{10}$$

Esta ecuación contiene configuraciones similares para las variables independientes (X) y sus coeficientes beta (β) que la regresión lineal. No obstante, hay diferencias clave:

1. En regresión logística, $\hat{Y}_i$ representa la probabilidad estimada de estar en una categoría de resultado binario (por ejemplo, tener la enfermedad) frente a no estar en ella, en lugar de un resultado continuo estimado.

2. La expresión $e^{\beta_0+\beta_1 X_1+\beta_2 X_2+\ldots+\beta_i X_i}$ representa la ecuación de regresión lineal para las variables independientes expresadas en escala logit, y no en el formato lineal original.

Esta transformación a escala logit es esencial en el modelo de regresión logística, ya que un resultado binario expresado como probabilidad debe estar entre 0 y 1. En cambio, las variables independientes podrían asumir cualquier valor. Si no se rectifica esta discrepancia, los valores predichos del modelo podrían caer fuera del rango de 0 a 1. La escala logit resuelve este problema al transformar matemáticamente la ecuación original de regresión lineal para producir el logit o logaritmo natural de las probabilidades de estar en una categoría ($\hat{Y}$) frente a la otra ($1 - \hat{Y}$):

$$\ln\left(\frac{\hat{Y}}{1-\hat{Y}}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_i X_i \tag{11}$$

En el contexto de estas ecuaciones, la regresión logística identifica, mediante ciclos iterativos, la combinación lineal más fuerte de variables independientes que aumente la probabilidad de detectar el resultado observado—un proceso conocido como estimación por máxima verosimilitud.

Introducción

El artículo revisa detalladamente el modelo de regresión logística (RL), una técnica multivariable esencial para analizar relaciones entre variables independientes y una variable dependiente categórica, especialmente en investigaciones médicas. A través del análisis de 37 artículos científicos publicados entre 2000 y 2018 y seis libros de texto especializados, los autores exploran conceptos clave de la RL, su aplicación, problemas frecuentes y propuestas para mejorar su uso.

Antecedentes Históricos de la Regresión Logística

La regresión logística tiene sus raíces en el siglo XIX, con los trabajos de Pierre François Verhulst, quien introdujo la *curva logística* para modelar el crecimiento poblacional. Sin embargo, fue en el siglo XX cuando su aplicación estadística tomó forma. En 1944, Joseph Berkson introdujo el *modelo logit* en el contexto de bioestadística, proponiéndolo como alternativa al modelo probit. La RL fue adoptada ampliamente en estudios biomédicos a partir de la década de 1960, gracias a su capacidad para manejar variables dicotómicas y ofrecer interpretaciones claras a través del odds ratio. En décadas recientes, la regresión logística se ha convertido en una herramienta fundamental para el análisis de datos en epidemiología, medicina clínica, y ciencias sociales.

Fundamentos del Modelo de Regresión Logística

La RL es ideal para predecir la probabilidad de ocurrencia de un evento binario (sí/no) y se basa en la transformación logística del *odds ratio* (razón de probabilidades). A diferencia de la regresión lineal, no requiere que las variables independientes sigan una distribución normal ni que la relación con la dependiente sea lineal.

La función logística

La función logística transforma la probabilidad de un evento en odds, y posteriormente en log-odds (*logit*), acotando los valores entre 0 y 1. Esto asegura interpretaciones coherentes para eventos dicotómicos. El modelo toma la forma:

$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_k X_k$$

Selección de variables

Se discuten criterios para seleccionar variables dependientes (ej. enfermedad/salud) y predictoras (factores clínicos), advirtiendo sobre el impacto negativo del sesgo de selección, multicolinealidad y tamaño de muestra reducido. El artículo destaca la importancia del conocimiento previo para seleccionar variables relevantes y evitar el sobreajuste.

Evaluación del Modelo

Evaluación general

Se emplean dos pruebas principales:

- **Razón de verosimilitudes (likelihood ratio test)**: compara un modelo completo con uno nulo, evaluando si los predictores mejoran significativamente la predicción.
- **Prueba de Hosmer-Lemeshow**: mide el ajuste entre valores observados y esperados por deciles de riesgo. Un valor $p > 0{,}05$ indica buen ajuste.

Evaluación de predictores

La significancia individual de cada predictor se evalúa con el **estadístico Wald**, basado en la relación entre el coeficiente estimado y su error estándar. También se puede usar la razón de verosimilitudes para cada predictor.

Exactitud Predictiva y Discriminación

- **Tabla de clasificación**: compara predicciones contra observaciones reales, generando métricas como sensibilidad, especificidad, precisión y valor predictivo.
- **Curva ROC (Receiver Operating Characteristic)**: representa la sensibilidad frente a 1 - especificidad. El área bajo la curva (AUC) cuantifica la capacidad discriminativa del modelo. Un AUC de 0.5 indica clasificación aleatoria, mientras que 1.0 representa clasificación perfecta.

Validación del Modelo

Se destaca la importancia de validar los modelos, ya sea de manera **interna** (con subconjuntos del mismo conjunto de datos) o **externa** (con nuevos datos). Se discuten métodos como *bootstrap*, *jackknife* y validación cruzada.

También se mencionan medidas como:

- R^2 **de Cox & Snell**
- R^2 **de Nagelkerke**

Estas proporcionan información sobre el poder explicativo del modelo, aunque no son equivalentes al R^2 clásico de regresión lineal.

Aplicación Práctica

Se presenta un estudio de caso usando RL para investigar factores que influyen en la decisión de partos por cesárea en mujeres embarazadas. Se evidenció que:

- El peso del bebé menor a 3.5 kg reduce la probabilidad de cesárea.
- Mujeres con más de tres partos tienen menor probabilidad de requerir cesárea.

El modelo mostró buen ajuste (Hosmer-Lemeshow $p = 0{,}65$), alta capacidad explicativa (R^2 de Nagelkerke = 0.723) y precisión predictiva del 82.9 %.

Conclusiones

Los autores concluyen que, aunque la RL es una herramienta poderosa, su uso en la investigación médica presenta deficiencias notables:

- Tamaños de muestra insuficientes.
- Falta de validación del modelo.
- Reportes incompletos sobre el ajuste y la significancia de predictores.

Recomiendan mayor rigor metodológico y transparencia en los reportes, así como comparar RL con métodos más recientes como redes neuronales o árboles de decisión en futuras investigaciones.

Resumen

Las técnicas de regresión son versátiles en su aplicación a la investigación médica, ya que permiten medir asociaciones, predecir resultados y controlar efectos de variables de confusión. Como una de estas técnicas, la regresión logística representa una forma eficiente y poderosa de analizar el efecto de un grupo de variables independientes sobre un resultado binario, cuantificando la contribución única de cada variable. Usando componentes de la regresión lineal reflejados en la escala logit, la regresió...

Consideraciones importantes al aplicar regresión logística incluyen: la selección adecuada de variables independientes, el cumplimiento de los supuestos necesarios, y la elección de una estrategia adecuada de construcción del modelo. La selección de variables debe guiarse por teorías aceptadas, investigaciones empíricas previas, consideraciones clínicas y análisis estadísticos univariados, incluyendo el reconocimiento de posibles variables de confusión.

Entre los supuestos básicos que deben cumplirse se encuentran: independencia de errores, linealidad en la escala logit para variables continuas, ausencia de multicolinealidad y falta de valores atípicos con fuerte influencia. Además, debe asegurarse un número adecuado de eventos por variable para evitar el sobreajuste, recomendándose comúnmente entre 10 y 20 eventos por covariable.

En cuanto a las estrategias de modelado, existen tres tipos generales: directa/estándar, secuencial/jerárquica y por pasos/estadística, cada una con distinto énfasis y propósito. Antes de extraer conclusiones definitivas, se recomienda cuantificar formalmente la validez interna del modelo (es decir, su replicabilidad en el mismo conjunto de datos) y su validez externa (generalización a otros conjuntos de datos).

El ajuste general del modelo al conjunto de datos se evalúa mediante diversas medidas de bondad de ajuste, siendo preferible un menor error entre los valores observados y los predichos. También se recomienda el uso de estadísticas diagnósticas para valorar la adecuación del modelo. Finalmente, los resultados para las variables independientes se reportan típicamente como razones de momios (odds ratios, ORs) con intervalos de confianza al 95 %.

La regresión logística es una forma eficiente y poderosa de analizar el efecto de un grupo de variables independientes sobre un resultado binario cuantificando la contribución única de cada variable independiente, por otra parte la regresión logística identifica iterativamente la combinación lineal más fuerte de variables con la mayor probabilidad de detectar el resultado observado. La regresión logística tiene sus raíces en el siglo XIX, con los trabajos de Pierre François Verhulst, quien introdujo la *curva logística* para modelar el crecimiento poblacional. Sin embargo, fue en el siglo XX cuando su aplicación estadística tomó forma. En 1944, Joseph Berkson introdujo el *modelo logit* en el contexto de bioestadística, proponiéndolo como alternativa al modelo probit. La Regresión Logística fue adoptada ampliamente en estudios biomédicos a partir de la década de 1960, gracias a su capacidad para manejar variables dicotómicas y ofrecer interpretaciones claras a través del odds ratio. En décadas recientes, la regresión logística se ha convertido en una herramienta fundamental para el análisis de datos en epidemiología, medicina clínica, y ciencias sociales.

La regresión logística es ideal para predecir la probabilidad de ocurrencia de un evento binario (sí/no) y se basa en la transformación logística del *odds ratio* (razón de probabilidades). A diferencia de la regresión lineal, no requiere que las variables independientes sigan una distribución normal ni que la relación con la dependiente sea lineal.

Es importante considerar en una regresión logística la *selección de variables independientes* para esto uno debe guiarse por factores tales como teoría existente, investigaciones empíricas previas, consideraciones clínicas y análisis estadísticos univariados, reconociendo las posibles variables de confusión que deben ser consideradas.

Los supuestos básicos que deben cumplirse para la regresión logística incluyen

- independencia de errores,
- linealidad en el *logit* para variables continuas,
- ausencia de multicolinealidad y
- falta de valores atípicos fuertemente influyentes. Adicionalmente,
- existencia de un número adecuado de eventos por variable independiente para evitar un modelo sobreajustado, con un mínimo comúnmente recomendado de "reglas prácticas" que van de 10 a 20 eventos por covariable.

Respecto a las estrategias de construcción de modelos, los tres tipos generales son:

- directa/estándar,
- secuencial/jerárquica y
- por pasos/estadística,

cada uno con un énfasis y propósito diferente. Antes de llegar a conclusiones definitivas a partir de los resultados de cualquiera de estos métodos, se debe cuantificar formalmente la validez interna del modelo (es decir, su replicabilidad dentro del mismo conjunto de datos) y su validez externa (es decir, su generalizabilidad más allá de la muestra actual).

El ajuste general del modelo de regresión logística a los datos de muestra se evalúa utilizando varias *medidas de bondad de ajuste*, donde un mejor ajuste se caracteriza por una menor diferencia entre los valores observados y los valores predichos por el modelo. También se recomienda el uso de *estadísticas de diagnóstico* para evaluar aún más la adecuación del modelo. Finalmente, los resultados para las variables independientes suelen reportarse como *razones de momios* (odds ratios, ORs) con *intervalos de confianza* (IC) del 95 %.

63. Importancia de la regresión en investigación

La **regresión** es un método valioso de investigación debido a su versátil aplicación en diferentes contextos de estudio. Por ejemplo, se puede utilizar para examinar asociaciones entre un resultado y varias variables independientes (también comúnmente conocidas como covariables, predictores o variables explicativas)[**?**], o para determinar qué tan bien puede predecirse un resultado a partir de un conjunto de variables independientes[**?**, **?**]. Adicionalmente, uno puede estar interesado en controlar el efecto de variables independientes específicas, particularmente aquellas que actúan como variables de confusión (es decir, cuya relación tanto con el resultado como con otra variable independiente oscurece la relación entre esa variable independiente y el resultado)[**?**, **?**]. Esta última aplicación es especialmente útil en contextos donde no es posible asignar aleatoriamente sujetos a grupos de tratamiento, como sucede en investigaciones observacionales. Con asignación aleatoria, normalmente se puede ejercer un control adecuado sobre las variables de confusión, ya que los grupos aleatorizados tienden a tener una distribución equitativa o balanceada de dichas variables[**?**].

64. Regresión Logística

Existen diferentes tipos de regresión, dependiendo de los objetivos de investigación y del formato de las variables, siendo la regresión lineal una de las más utilizadas. La *regresión lineal* analiza resultados continuos (es decir, aquellos que pueden sumarse, restarse, multiplicarse o dividirse de manera significativa, como el peso) y asume que la relación entre el resultado y las variables independientes sigue una forma funcional determinada. Sin embargo, generalmente es más deseable determinar la influencia de múltiples factores al mismo tiempo, ya que de este modo se pueden observar las contribuciones únicas de cada variable después de controlar por los efectos de las demás. En este caso, la regresión lineal multivariada es la opción adecuada.

La ecuación básica para la regresión lineal con múltiples variables independientes es:

$$\hat{Y} = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_i X_i. \tag{12}$$

Los componentes de esta ecuación son los siguientes:

- $\hat{Y}$ es el resultado continuo estimado.
- $\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_i X_i$ es la ecuación de regresión lineal para las variables independientes del modelo, donde:
  - β_0 es la ordenada al origen o punto en el que la línea de regresión toca el eje vertical Y. Se considera un valor constante.
  - $\beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_i X_i$ es el valor de cada variable independiente (X_i) ponderado por su respectivo coeficiente beta (β). Los coeficientes beta determinan la pendiente de la línea de regresión, cuanto mayor sea el coeficiente beta, más fuerte es la contribución de dicha variable al resultado.

Para una variable binaria, como la mortalidad, la regresión logística es el método usualmente elegido, la regresión logística puede incluir una o múltiples variables independientes, aunque examinar múltiples variables es generalmente más informativo, ya que permite revelar la contribución única de cada variable ajustando por las demás. La regresión logística tiene ecuación:

$$\text{Probabilidad del resultado}(\hat{Y}_i) = \frac{e^{\beta_0+\beta_1 X_1+\beta_2 X_2+\ldots+\beta_i X_i}}{1 + e^{\beta_0+\beta_1 X_1+\beta_2 X_2+\ldots+\beta_i X_i}}. \tag{13}$$

65. Transformación logística y elección de variables independientes

Un aspecto importante de la regresión logística es que conserva muchas características de la regresión lineal en su análisis de resultados binarios. Sin embargo, existen diferencias clave entre las dos ecuaciones:

1. $\hat{Y}_i$ representa la probabilidad estimada de pertenecer a una de las dos categorías binarias del resultado (categoría i) en lugar de representar un resultado continuo estimado.
2. $e^{\beta_0+\beta_1 X_1+\beta_2 X_2+\ldots+\beta_i X_i}$ representa la ecuación de regresión lineal para las variables independientes expresadas en la escala *logit*.

La razón de esta transformación *logit* radica en los parámetros básicos del modelo de regresión logística, un resultado binario expresado como probabilidad debe estar entre 0 y 1 [**?**]. La escala logit resuelve este problema al transformar matemáticamente la ecuación de regresión lineal original para producir el logit (o logaritmo natural) de las razones de momios (odds) de estar en una categoría ($\hat{Y}$) frente a la otra categoría ($1 - \hat{Y}$):

$$logit(\hat{Y}) = \ln\left(\frac{\hat{Y}}{1-\hat{Y}}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_i X_i \tag{14}$$

En el contexto de estas ecuaciones, la regresión logística identifica, mediante ciclos iterativos, la combinación lineal más fuerte de variables independientes que aumente la probabilidad de detectar el resultado observado —un proceso conocido como estimación de máxima verosimilitud [**?**, **?**]. Para asegurar que la regresión logística produzca un modelo preciso, se deben considerar factores críticos como la selección de variables independientes y la elección de la estrategia de construcción del modelo.

65.1. Variables independientes

Criterio 1. ***[Criterio de selección]*** *Es muy importante seleccionar correctamente las variables independientes. Aunque la regresión logística es bastante flexible y permite distintos tipos de variables (continuas, ordinales y categóricas), siempre debe justificarse la selección de variables utilizando: teoría bien establecida, investigaciones previas, observaciones clínicas, análisis estadístico preliminar, o una combinación razonada de estas opciones.*

*Alternativamente, uno podría optar por incluir todas las variables independientes relevantes independientemente de sus resultados univariados, ya que puede haber variables clínicamente importantes que merezcan inclusión a pesar de su desempeño estadístico; sin embargo, incluir demasiadas variables independientes en el modelo puede conducir a un resultado matemáticamente inestable, con menor capacidad de generalización más allá de la muestra actual del estudio [**?**, **?**].*

*Una parte clave del proceso de selección de variables es reconocer y considerar el papel de los posibles factores de confusión. Como se describió previamente, las variables de confusión son aquellas cuya relación tanto con el resultado como con otra variable independiente oculta la verdadera asociación entre esa variable independiente y el resultado [**?**, **?**].*

Independientemente del método para seleccionar las variables independientes, deben cumplirse ciertos supuestos básicos al aplicar regresión logística.

Supuesto 1. ***[Independencia de los errores]*** *Todos los resultados del grupo de muestra deben ser independientes entre sí; si los datos incluyen mediciones repetidas u otros resultados correlacionados, los errores también estarán correlacionados y el supuesto se violará.[**?**] Existen otros métodos para analizar datos correlacionados mediante técnicas de regresión logística, consultar a Stokes et al.,[**?**] Newgard et al.,[**?**, **?**] y Allison.[**?**].*

Supuesto 2. ***[Linealidad en el logit]*** *para las variables continuas independientes, debe existir una relación lineal entre estas variables y sus respectivos resultados transformados en logit. Esto se puede realizar a través de la creación de un término de interacción entre cada variable continua independiente y su logaritmo natural. Si alguno de estos términos es estadísticamente significativo, se considera que el supuesto no se cumple [**?**, **?**]. Las soluciones incluyen codificación dicotómica de la variable independiente,[**?**] o su transformación estadística a otra escala [**?**, **?**].*

Supuesto 3. ***[Ausencia de multicolinealidad]****, o redundancia entre variables independientes, un modelo de regresión logística con variables independientes altamente correlacionadas usualmente genera errores estándar grandes para los coeficientes beta (o pendientes) estimados. La solución común es eliminar una o más variables redundantes.[?]*

Supuesto 4. ***[Ausencia de valores atípicos altamente influyentes]****, es decir, casos en los que el resultado predicho para un miembro de la muestra difiere considerablemente de su valor real, si hay demasiados valores atípicos, la precisión general del modelo puede verse comprometida. La detección de valores atípicos se realiza examinando los residuales (diferencia entre los valores predichos y los resultados reales) junto con estadísticas diagnósticas y gráficas [?, ?]; luego, se puede comparar el ajuste general del modelo y los coeficientes beta estimados con y sin los casos atípicos, dependiendo de la magnitud del cambio, uno podría conservar los valores atípicos cuyo efecto no sea dramático[?] o eliminar aquellos con una influencia particularmente fuerte sobre el modelo [?, ?].*

Criterio 2. ***[Número de variables a incluir]*** *Como parte del proceso de selección de qué variables independientes incluir, también se debe decidir cuántas. El reto es seleccionar el menor número posible de variables independientes que expliquen mejor el resultado sin descuidar las limitaciones del tamaño de muestra [?, ?]. En términos generales, un modelo sobreajustado tiene coeficientes beta estimados para las variables independientes mucho mayores de lo que deberían ser, además de errores estándar más altos de lo esperado [?]. Este tipo de situación genera inestabilidad en el modelo porque la regresión logística requiere más resultados que variables independientes para poder iterar soluciones diferentes en busca del mejor ajuste a través del método de máxima verosimilitud [?, ?].*

Aunque no existe un estándar universalmente aceptado, hay algunas reglas generales derivadas en parte de estudios de simulación. Una de estas reglas sugiere que por cada variable independiente, debe haber al menos 10 resultados por cada categoría binaria, siendo el resultado menos frecuente el que determina el número máximo de variables independientes [?, ?]. Algunos estadísticos recomiendan una regla general aún más estricta de 20 resultados por variable independiente, dado que una relación más alta tiende a mejorar la validez del modelo[?].

65.2. Estrategias de Construcción del Modelo

Además de la cuidadosa selección de las variables independientes, se debe elegir el tipo adecuado de modelo de regresión logística para el estudio. De hecho, seleccionar una estrategia de construcción del modelo está estrechamente relacionado con la elección de variables independientes, por lo que estos dos componentes deben considerarse simultáneamente al planear un análisis de regresión logística.

Existen tres enfoques generales para la construcción del modelo que se aplican a las técnicas de regresión en general, cada uno con un énfasis y propósito diferente:

- **Directo** (es decir, completo, estándar o simultáneo): Este enfoque es una especie de valor por defecto, ya que introduce todas las variables independientes en el modelo al mismo tiempo y no hace suposiciones sobre el orden o la importancia relativa de dichas variables [?, ?]. El enfoque directo es más adecuado si no existen hipótesis previas sobre cuáles variables tienen mayor relevancia que otras.

- **Secuencial** (es decir, jerárquico): las variables se añaden secuencialmente para evaluar si mejoran el modelo de acuerdo a un orden predeterminado de prioridad [**?**, **?**]. Aunque este enfoque es útil para clarificar patrones causales entre variables independientes y resultados, puede volverse complejo conforme aumentan los patrones causales, dificultando así la obtención de conclusiones definitivas sobre los datos en algunos casos [**?**].

- **Paso a paso** (es decir, estadístico): En contraste con los dos métodos anteriores, la regresión paso a paso identifica variables independientes que deben mantenerse o eliminarse del modelo con base en criterios estadísticos predefinidos que están influenciados por las características únicas de la muestra analizada [**?**, **?**]. Existen distintos tipos de técnicas paso a paso, incluyendo selección hacia adelante y eliminación hacia atrás con una contribución no significativa al resultado son eliminadas una por una hasta que sólo queden las variables estadísticamente significativas.[**?**, **?**] Otra estrategia de construcción del modelo que es conceptualmente similar a la regresión por pasos se llama *selección del mejor subconjunto'*, en la que se comparan modelos separados con diferentes números de variables independientes para determinar el mejor ajuste [**?**]

Estas estrategias de construcción no son necesariamente intercambiables, ya que pueden producir diferentes medidas de ajuste del modelo y diferentes estimaciones puntuales para las variables independientes a partir de los mismos datos. Por lo tanto, identificar el modelo apropiado para los objetivos del estudio es extremadamente importante.

Nota 18. *Aunque la regresión por pasos se usa frecuentemente en la investigación clínica, su uso es algo controvertido porque se basa en una selección automatizada de variables que tiende a aprovechar factores aleatorios en una muestra dada. Además, la regresión por pasos puede producir modelos que no parecen completamente razonables desde una perspectiva biológica. Ante estas preocupaciones, algunos argumentan que la regresión por pasos se reserva mejor para el tamizaje preliminar o únicamente para pruebas de hipótesis, como en casos de resultados novedosos y una comprensión limitada de las contribuciones de las variables independientes. Sin embargo, otros señalan que los métodos por pasos no son en sí el problema (y de hecho pueden ser bastante efectivos en ciertos contextos); en cambio, el verdadero problema es una interpretación descuidada de los resultados sin valorar completamente los pros y contras de este enfoque. Por tanto, si uno elige crear un modelo por pasos, es importante validar posteriormente los resultados antes de sacar conclusiones. No obstante, debe destacarse que todos los tipos de modelos requieren validación formal antes de que se consideren definitivos para uso futuro, ya que se espera que los modelos funcionen mejor con la muestra original que con muestras subsiguientes.[**?**, **?**, **?**]*

65.3. Validación Interna y Externa del Modelo

Al validar modelos de regresión logística, existen numerosos métodos entre los cuales elegir, cada uno más o menos apropiado según los parámetros del estudio como el tamaño de muestra. Para establecer la validez interna (confirmación de resultados del modelo con el mismo conjunto de datos), los métodos comunes incluyen:

- **método de retención, o división de la muestra en dos subgrupos** antes de la construcción del modelo, con el grupo de *entrenamiento* usado para crear el modelo de regresión logística y el grupo de *prueba* usado para validarlo; [**?**, **?**]
- **validación cruzada k-fold o división de la muestra en k subgrupos de igual tamaño** para propósitos de entrenamiento y validación;[**?**]
- **validación cruzada *uno fuera* (leave-one-out)**, una variante del método k-fold donde el número de particiones es igual al número de sujetos en la muestra;[**?**] y
- **bootstrapping** es decir, obtener submuestras repetidas con reemplazo de toda la muestra [**?**, **?**].

Además de validar internamente el modelo, uno debería intentar validarlo externamente en un nuevo entorno de estudio como una prueba adicional de su viabilidad estadística y utilidad clínica [**?**, **?**]. Si los resultados de la validación interna o externa presentan alguna alerta se recomienda hacer ajustes al modelo según sea necesario, o definir explícitamente cualquier restricción para el uso futuro del modelo.[**?**]

65.4. Interpretación de los Resultados del Modelo

- **Evaluación del Ajuste General del Modelo.** Una vez que se ha creado el modelo de regresión logística, se determina qué tan bien se ajusta a los datos de la muestra en su totalidad. Dos de los métodos más comunes para evaluar el ajuste del modelo son la prueba de chi-cuadrado de Pearson y la desviación residual. Ambas miden la diferencia entre los resultados observados y los resultados predichos por el modelo, donde un mal ajuste del modelo se indica mediante valores de prueba elevados, lo que señala una diferencia mayor [**?**, **?**, **?**].

  Otra medida comúnmente utilizada del ajuste del modelo es la prueba de bondad de ajuste de *Hosmer-Lemeshow*, que divide a los sujetos en grupos iguales (a menudo de 10) según su probabilidad estimada del resultado. El decil más bajo está compuesto por aquellos que tienen menor probabilidad de experimentar el resultado. Si el modelo tiene buen ajuste, los sujetos que experimentaron el resultado principal caerán en su mayoría en los deciles de mayor riesgo. Un modelo con mal ajuste resultará en sujetos distribuidos de manera más uniforme a lo largo de los deciles de riesgo para ambos resultados binarios [**?**, **?**].

  Las ventajas de las pruebas de Hosmer-Lemeshow incluyen su aplicación sencilla y facilidad de interpretación, las limitaciones incluyen la dependencia de las pruebas sobre cómo se definen los puntos de corte de los grupos y los algoritmos computacionales utilizados, así como una menor capacidad para identificar modelos con mal ajuste en ciertas circunstancias. Otras alternativas menos comunes para evaluar el ajuste del modelo son descritas por Hosmer et al [**?**] y Kuss [**?**].

  Otra opción para ampliar los resultados del ajuste del modelo y de las estadísticas diagnósticas, es evaluando la capacidad del modelo para discriminar entre grupos. Las formas comunes de hacer esto incluyen

1. Tablas de clasificación, donde la pertenencia a un grupo dentro de una categoría binaria del resultado se predice usando probabilidades estimadas y puntos de corte predefinidos, y
2. Área bajo la curva característica operativa del receptor (AUROC), donde un valor de 0.5 significa que el modelo no es mejor que el azar para discriminar entre los sujetos que tienen el resultado y los que no, y un valor de 1.0 indica que el modelo discrimina perfectamente entre sujetos. *El AUROC se usa a menudo cuando se desean considerar diferentes puntos de corte para la clasificación y así maximizar tanto la sensibilidad como la especificidad* [**?**].

- **Interpretación de los Resultados de Variables Individuales.** Las variables independientes usualmente se presentan como razones de momios (ORs, por sus siglas en inglés), que revelan la fuerza de la contribución de la variable independiente al resultado y se definen como las probabilidades de que ocurra el resultado ($\hat{Y}$) frente a que no ocurra, $(1 - \hat{Y})$, para cada variable independiente. La relación entre la razón de momios (OR) y el coeficiente beta estimado de la variable independiente se expresa como $\text{OR} = e^{\beta_i}$. Con base en esta fórmula, un cambio de una unidad en la variable independiente multiplica la probabilidad del resultado por la cantidad contenida en e^{β_i}.

  Para un modelo de regresión logística con solo una variable independiente, la OR se considera "no ajustada" porque no hay otras variables cuya influencia deba ser ajustada o restada. En contraste, si el modelo de regresión logística incluye múltiples variables independientes, las OR ahora son *ajustadas* porque representan la contribución única de la variable independiente después de ajustar (o restar) los efectos de las otras variables en el modelo, en conclusión las OR ajustadas suelen ser menores que sus contrapartes no ajustadas. Interpretar las OR también depende de si la variable independiente es continua o categórica. Para las variables continuas, primero se debe identificar una unidad de medida significativa que exprese mejor el grado de cambio en el resultado asociado con esa variable independiente. Finalmente, los intervalos de confianza (IC) al 95 % se informan rutinariamente junto con las OR como una medida de precisión (es decir, si los hallazgos probablemente se mantendrán en la población no observada). Si el IC cruza 1.00, es posible que no haya una diferencia significativa en esa población.

66. Fundamentos del Modelo de Regresión Logística

- La función logística transforma la probabilidad de un evento en **odds**, y posteriormente en **log-odds** (*logit*), acotando los valores entre 0 y 1. Esto asegura interpretaciones coherentes para eventos dicotómicos. El modelo toma la forma:

$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_k X_k$$

- **Evaluación general** Se emplean dos pruebas principales:

- **Razón de verosimilitudes (likelihood ratio test)**: compara un modelo completo con uno nulo, evaluando si los predictores mejoran significativamente la predicción.
- **Prueba de Hosmer-Lemeshow**: mide el ajuste entre valores observados y esperados por deciles de riesgo. Un valor $p > 0{,}05$ indica buen ajuste.

- **Evaluación de predictores** La significancia individual de cada predictor se evalúa con el **estadístico Wald**, basado en la relación entre el coeficiente estimado y su error estándar. También se puede usar la razón de verosimilitudes para cada predictor.

- **Exactitud Predictiva y Discriminación**

  - **Tabla de clasificación**: compara predicciones contra observaciones reales, generando métricas como sensibilidad, especificidad, precisión y valor predictivo.
  - **Curva ROC (Receiver Operating Characteristic)**: representa la sensibilidad frente a 1 - especificidad. El área bajo la curva (AUC) cuantifica la capacidad discriminativa del modelo. Un AUC de 0.5 indica clasificación aleatoria, mientras que 1.0 representa clasificación perfecta.

- **Validación del Modelo**

  Se destaca la importancia de validar los modelos, ya sea de manera **interna** (con subconjuntos del mismo conjunto de datos) o **externa** (con nuevos datos). Se discuten métodos como *bootstrap*, *jackknife* y validación cruzada.

  También se mencionan medidas como:

  - R^2 **de Cox & Snell**
  - R^2 **de Nagelkerke**

  Estas proporcionan información sobre el poder explicativo del modelo, aunque no son equivalentes al R^2 clásico de regresión lineal.

Para una variable binaria $y_i \in \{0, 1\}$, el modelo de regresión logística predice la probabilidad:

$$p_i = \frac{1}{1 + e^{-x_i \beta}}$$

Función de verosimilitud

$$L(\beta) = \prod_{i=1}^{n} p_i^{y_i} (1 - p_i)^{1 - y_i}$$

Log-verosimilitud

$$\ell(\beta) = \sum_{i=1}^{n} \left[y_i \ln(p_i) + (1 - y_i) \ln(1 - p_i) \right]$$

Derivadas

Gradiente:

$$\nabla_\beta \ell(\beta) = X^\top (y - p)$$

Hessiano (segunda derivada):

$$\nabla^2_\beta \ell(\beta) = -X^\top V X, \quad V = \mathrm{diag}(p_i(1 - p_i))$$

Regularización Ridge (L2)

Se añade un término penalizado:

$$\ell_\lambda(\beta) = \ell(\beta) - \frac{\lambda}{2}\|\beta\|^2$$

Gradiente regularizado:

$$\nabla_\beta \ell_\lambda(\beta) = X^\top (y - p) - \lambda\beta$$

Hessiano regularizado:

$$\nabla^2_\beta \ell_\lambda(\beta) = -X^\top V X - \lambda I$$

TR-IRLS (Trust Region IRLS)

Actualización generalizada del paso de Newton:

$$\beta^{(k+1)} = \beta^{(k)} + s$$

Donde s es solución de:

$$(X^\top V X + \lambda I)s = X^\top V z - \lambda\beta$$

Eventos raros y correcciones

Ajuste del intercepto (King & Zeng)

$$\tilde{\beta}_0 = \hat{\beta}_0 - \ln\left[\left(\frac{1-\tau}{\tau}\right)\left(\frac{\hat{y}}{1-\hat{y}}\right)\right]$$

Muestreo estratificado

Peso para observación i:

$$w_i = \frac{Q_i}{H_i}$$

Verosimilitud ponderada:

$$\ell_w(\beta) = \sum_{i=1}^{n} w_i \ln\left(\frac{e^{x_i\beta}}{1 + e^{x_i\beta}}\right)$$

Corrección de Firth

Se basa en el ajuste de penalización de tipo Jeffreys:

$$\ell^*(\beta) = \ell(\beta) + \frac{1}{2}\log|I(\beta)|$$

Regla de decisión

$$\hat{y}_i = \begin{cases} 1 & \text{si } p_i \geq c \\ & \qquad\qquad (\text{usualmente } c = 0{,}5) \\ 0 & \text{si } p_i < c \end{cases}$$

Nota 19.
- *En problemas de clasificación con etiquetas binarias o etiquetas con una cantidad finita de opciones, la evaluación usualmente se realiza por medio de la matriz de confusión: el número de verdaderos/falsos positivos y negativos.*

	Positivo	*Negativo*
Predecido Positivo	*TP*	*FP*
Predicido Negativo	*FN*	*TN*

- *Para problemas de regresión con etiquetas de valores continuos usualmente se calcula la raíz del error cuadrático medio*

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2} \tag{15}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y})^2}{\sum_{i=1}^{N}(y_i - \overline{y})^2} \tag{16}$$

En cualquiera de los dos casos la evaluación final se lleva a cabo en el conjunto de prueba, el cuál es esencial dado que el último objetivo es obtener el predictor más general en los datos no utilizados para entrenar el algoritmo.

Nota 20. *Las siguientes métricas se utilizan para medir el rendimiento de un modelo en función de su capacidad para predecir correctamente las clases de un conjunto de datos.*

- ***Recall (Recall o Sensibilidad):*** *Conocido como sensibilidad o tasa positiva real, mide la capacidad de un modelo para identificar correctamente todos los ejemplos positivos en un conjunto de datos. Se calcula como el número de verdaderos positivos dividido por la suma de verdaderos positivos y falsos negativos:*

$$Recall = \frac{Verdaderos\ Positivos}{Verdaderos\ Positivos + Falsos\ Negativos} \tag{17}$$

Un recall alto significa que el modelo es bueno para detectar los casos positivos, minimizando los falsos negativos. Es importante en situaciones donde los falsos negativos son costosos o críticos.

- *Precision (Precisión): La precisión mide la capacidad de un modelo para predecir correctamente los casos positivos entre todas las predicciones positivas que realiza. Se calcula como el número de verdaderos positivos dividido por la suma de verdaderos positivos y falsos positivos:*

$$Precision = \frac{Verdaderos\ Positivos}{Verdaderos\ Positivos + Falsos\ Positivos} \tag{18}$$

Una alta precisión significa que el modelo tiene una baja tasa de falsos positivos, es decir, que cuando predice una clase como positiva, es probable que sea correcta. La precisión es importante en situaciones en las que los falsos positivos son costosos o no deseados.

- *Specificity (Especificidad): La especificidad mide la capacidad de un modelo para predecir correctamente los casos negativos entre todas las predicciones negativas que realiza. También se conoce como tasa negativa real. Se calcula como el número de verdaderos negativos dividido por la suma de verdaderos negativos y falsos positivos:*

$$Specificity = \frac{Verdaderos\ Negativos}{Verdaderos\ Negativos + Falsos\ Positivos} \tag{19}$$

Una alta especificidad indica que el modelo es bueno para identificar correctamente los casos negativos, minimizando los falsos positivos. Esto es importante en situaciones en las que los falsos positivos son costosos o problemáticos.

Estas métricas proporcionan una forma más completa de evaluar el rendimiento de un modelo de clasificación que simplemente mirar la precisión general.

En la ingeniería de proteínas, las similitudes en secuencias en ambos subconjuntos de datos deben ser tenidas en cuenta. Si alguna familia de proteínas está sobre representada en el conjunto de prueba, el predictor resultante puede resultar sesgado hacia la identificación de patrones válidos solamente para esta familia. Si algunas secuencias en el conjunto de prueba son muy cercanas al conjunto de entrenamiento, la evaluación final de desempeño dara resultados sobre optimistas.

En el paso 2 de entrenamiento, es posible ajustar el predictor o seleccionar de entre varios predictores, usualmente por medio de validación $k - fold$. En este caso los datos de entrenamiento se subdividen en K subconjuntos y el flujo de trabajo se repite K veces, con cada uno de ellos utilizaods para la evaluación de los $K - 1$ subconjuntos utilizados para entrenar. El reto principal en el paso 2 para cualquiern entrenamiento tipo ML supervisado es evitar el subajuste de los datos (sesgo alto) y el sobre ajuste (varianza grande).

La **subestimación** ocurre cuando un predictor falla en encontrar patrones incluso en los datos de entrenamiento (cuando un modelo lineal simple se utiliza para explicar dependencia dependencias no lineales en los datos). El **sobreajuste** ocurre cuando el desempeño de un

predictor disminuye notablemente en los datos de prueba en comparación con los datos de prueba, debido al aprendizaje de demasiado detalle y ruido, en lugar de identificar patrones generales. Tanto el subajuste como el sobreajuste pueden ser debido a la insuficiente calidad de los datos: ruido excesivo, características faltantes o irrelevantes, sesgo en los datos, o datos dispersos. También pueden ocurrir como consecuencia de una pobre aplicación del algoritmo: excesiva o insuficiente flexibilidad en la selección de los parámetros, protocolo de entrenamiento inapropiado, o contaminación de los datos de entrenamiento con el conjunto de datos de prueba.

Dado un conjunto de datos (x_i, y_i) para $i = 1, \ldots, n$, queremos encontrar los coeficientes β_0 y β_1 que minimicen el error cuadrático:

$$\sum_{i=1}^{n}(y_i - (\beta_0 + \beta_1 x_i))^2$$

Es más sencillo resolver este problema usando la forma *centralizada*:

$$y_i = \beta_0^* + \beta_1(x_i - \bar{x}) + \varepsilon_i$$

donde:

$$\beta_0 = \beta_0^* - \beta_1\bar{x}$$

Paso 1: Derivada parcial respecto a β_0^*

Queremos minimizar:

$$L(\beta_0^*, \beta_1) = \sum_{i=1}^{n}[y_i - (\beta_0^* + \beta_1(x_i - \bar{x}))]^2$$

Calculamos la derivada parcial:

$$\begin{aligned}\frac{\partial L}{\partial \beta_0^*} &= \sum_{i=1}^{n} 2[y_i - (\beta_0^* + \beta_1(x_i - \bar{x}))](-1) \\ &= -2\sum_{i=1}^{n}[y_i - \beta_0^* - \beta_1(x_i - \bar{x})]\end{aligned}$$

Igualamos a cero:

$$\sum_{i=1}^{n}[y_i - \beta_0^* - \beta_1(x_i - \bar{x})] = 0$$

Distribuimos:

$$n\beta_0^* + \beta_1\sum_{i=1}^{n}(x_i - \bar{x}) = \sum_{i=1}^{n} y_i$$

Pero:

$$\sum_{i=1}^{n}(x_i - \bar{x}) = 0$$

Entonces:

$$n\beta_0^* = \sum_{i=1}^{n} y_i \Rightarrow \beta_0^* = \bar{y}$$

Paso 2: Derivada parcial respecto a β_1

Nuevamente:

$$L(\beta_0^*, \beta_1) = \sum_{i=1}^{n} [y_i - \beta_0^* - \beta_1(x_i - \bar{x})]^2$$

Derivamos con respecto a β_1:

$$\begin{aligned}\frac{\partial L}{\partial \beta_1} &= \sum_{i=1}^{n} 2[y_i - \beta_0^* - \beta_1(x_i - \bar{x})](-1)(x_i - \bar{x}) \\ &= -2\sum_{i=1}^{n} [y_i - \beta_0^* - \beta_1(x_i - \bar{x})](x_i - \bar{x})\end{aligned}$$

Igualamos a cero:

$$\sum_{i=1}^{n} [y_i - \beta_0^* - \beta_1(x_i - \bar{x})](x_i - \bar{x}) = 0$$

Sustituimos $\beta_0^* = \bar{y}$:

$$\sum_{i=1}^{n} [y_i - \bar{y} - \beta_1(x_i - \bar{x})](x_i - \bar{x}) = 0$$

Distribuimos:

$$\sum_{i=1}^{n} (y_i - \bar{y})(x_i - \bar{x}) - \beta_1 \sum_{i=1}^{n} (x_i - \bar{x})^2 = 0$$

Despejamos β_1:

$$\beta_1 = \frac{\sum_{i=1}^{n} (y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^{n} (x_i - \bar{x})^2} = \frac{S_{xy}}{S_{xx}}$$

Paso 3: Recuperar β_0

$$\beta_0 = \beta_0^* - \beta_1 \bar{x} = \bar{y} - \beta_1 \bar{x}$$

Para una variable binaria $y_i \in \{0, 1\}$, el modelo de regresión logística predice la probabilidad:

$$p_i = \frac{1}{1 + e^{-x_i \beta}}$$

Función de verosimilitud

$$L(\beta) = \prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{1-y_i}$$

Log-verosimilitud

$$\ell(\beta) = \sum_{i=1}^{n} \left[y_i \ln(p_i) + (1-y_i)\ln(1-p_i) \right]$$

Usando que $p_i = \frac{1}{1+e^{-x_i\beta}}$, entonces:

$$\ell(\beta) = \sum_{i=1}^{n} \left[y_i x_i \beta - \ln(1+e^{x_i\beta}) \right]$$

Derivadas: Gradiente y Hessiano

Gradiente

Partimos de:

$$\ell(\beta) = \sum_{i=1}^{n} \left[y_i x_i \beta - \ln(1+e^{x_i\beta}) \right]$$

Derivando con respecto a β_j:

$$\frac{\partial \ell(\beta)}{\partial \beta_j} = \sum_{i=1}^{n} \left[y_i x_{ij} - \frac{e^{x_i\beta}}{1+e^{x_i\beta}} x_{ij} \right] = \sum_{i=1}^{n} x_{ij}(y_i - p_i)$$

Forma vectorial del gradiente:

$$\nabla_\beta \ell(\beta) = X^\top (\mathbf{y} - \mathbf{p})$$

Hessiano

La derivada del gradiente es:

$$\frac{\partial^2 \ell(\beta)}{\partial \beta_j \partial \beta_k} = -\sum_{i=1}^{n} x_{ij} x_{ik} p_i (1-p_i)$$

Forma matricial:

$$\nabla^2_\beta \ell(\beta) = -X^\top V X, \quad \text{donde } V = \text{diag}(p_i(1-p_i))$$

Regularización Ridge (L2)

Penalización L2 añadida a la log-verosimilitud:

$$\ell_\lambda(\beta) = \ell(\beta) - \frac{\lambda}{2}\|\beta\|^2$$

Gradiente regularizado:

$$\nabla_\beta \ell_\lambda(\beta) = X^\top(\mathbf{y} - \mathbf{p}) - \lambda\beta$$

Hessiano regularizado:

$$\nabla^2_\beta \ell_\lambda(\beta) = -X^\top V X - \lambda I$$

TR-IRLS (Trust Region IRLS)

Actualización de Newton truncado:

$$\beta^{(k+1)} = \beta^{(k)} + s$$

Donde s resuelve:

$$(X^\top V X + \lambda I)s = X^\top V z - \lambda\beta$$

Eventos raros y correcciones

Ajuste del intercepto (King & Zeng)

$$\tilde{\beta}_0 = \hat{\beta}_0 - \ln\left[\left(\frac{1-\tau}{\tau}\right)\left(\frac{\hat{y}}{1-\hat{y}}\right)\right]$$

Muestreo estratificado y ponderación

Peso para observación i:

$$w_i = \frac{Q_i}{H_i}$$

Verosimilitud ponderada:

$$\ell_w(\beta) = \sum_{i=1}^{n} w_i \ln\left(\frac{e^{x_i\beta}}{1+e^{x_i\beta}}\right)$$

Corrección de Firth

Log-verosimilitud penalizada (ajuste de Jeffreys):

$$\ell^*(\beta) = \ell(\beta) + \frac{1}{2}\log|I(\beta)|$$

Regla de decisión

$$\hat{y}_i = \begin{cases} 1 & \text{si } p_i \geq c \\ 0 & \text{si } p_i < c \end{cases}, \quad \text{con } c = 0{,}5$$

Modelo base de regresión logística

La RL modela la probabilidad de un evento binario $y_i \in \{0, 1\}$ en función de un vector de predictores x_i mediante:

$$p_i = \frac{1}{1 + e^{-x_i \beta}} \tag{20}$$

La verosimilitud del modelo es:

$$L(\beta) = \prod_{i=1}^{n} p_i^{y_i} (1 - p_i)^{1 - y_i} \tag{21}$$

Y su log-verosimilitud:

$$\ell(\beta) = \sum_{i=1}^{n} [y_i \ln(p_i) + (1 - y_i) \ln(1 - p_i)] \tag{22}$$

Derivadas: Gradiente y Hessiano

- Gradiente:

$$\nabla_\beta \ell(\beta) = X^T (\mathbf{y} - \mathbf{p}) \tag{23}$$

- Hessiano:

$$\nabla^2_\beta \ell(\beta) = -X^T V X, \quad \text{donde } V = \text{diag}(p_i(1 - p_i)) \tag{24}$$

Regularización

Para evitar el sobreajuste, se añade un término de penalización L2 (ridge):

$$\ell_\lambda(\beta) = \ell(\beta) - \frac{\lambda}{2} \|\beta\|^2 \tag{25}$$

- Gradiente regularizado:

$$\nabla_\beta \ell_\lambda(\beta) = X^T (\mathbf{y} - \mathbf{p}) - \lambda \beta \tag{26}$$

- Hessiano regularizado:

$$\nabla^2_\beta \ell_\lambda(\beta) = -X^T V X - \lambda I \tag{27}$$

Algoritmo IRLS (Iteratively Reweighted Least Squares)

Una técnica común para estimar los parámetros del modelo es IRLS, que utiliza pesos $v_i = p_i(1 - p_i)$ y variables ajustadas z_i:

$$z_i = x_i\hat{\beta} + \frac{y_i - p_i}{v_i} \tag{28}$$

En cada iteración, se resuelve:

$$(X^T V X + \lambda I)\hat{\beta}^{(c+1)} = X^T V z^{(c)} \tag{29}$$

Este método es eficiente para bases de datos de tamaño moderado.

Algoritmo CG (Conjugate Gradient)

En problemas a gran escala, se recomienda el método del gradiente conjugado:

- Se inicializa el residuo $r^{(0)} = b - A\beta^{(0)}$.
- Se actualizan las direcciones de búsqueda y pasos óptimos iterativamente.
- Permite resolver sistemas lineales sin invertir matrices.

Es especialmente útil cuando $X^T V X$ es grande o disperso.

Correcciones para eventos raros

- **Ajuste del intercepto:** basado en la tasa real de eventos:

$$\tilde{\beta}_0 = \hat{\beta}_0 - \ln\left(\frac{1-\tau}{\tau} \cdot \frac{y}{1-y}\right) \tag{30}$$

- **Ponderación:** modifica la verosimilitud con pesos:

$$\ell(\beta|y, X) = \sum_{i=1}^{n} w_i \ln\left(\frac{e^{x_i\beta}}{1 + e^{x_i\beta}}\right) \tag{31}$$

Conclusiones clave

- La regresión logística es robusta y se adapta bien a diferentes contextos de datos.
- Las técnicas de regularización y los métodos numéricos como IRLS y CG la hacen escalable.
- Las correcciones para eventos raros mejoran la inferencia en muestras sesgadas.
- Es una herramienta base para modelos más complejos como regresión multinomial o clasificación ordinal.

67. Desarrollo Matemático

El modelo lineal univariado se expresa como:

$$h_\theta(x) = \theta_0 + \theta_1 x \tag{32}$$

Función de Costo (SSE - Error Cuadrático medio) está definida por:

$$J(\theta_0, \theta_1) = \frac{1}{2N} \sum_{i=1}^{N} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2 \tag{33}$$

Objetivo de Optimización: Nuestro objetivo es encontrar los valores óptimos de θ_0 y θ_1 que minimicen la función de costo

$$\min_{\theta_0, \theta_1} J(\theta_0, \theta_1) \tag{34}$$

Gradiente de la función de costo respecto a θ_0

$$\frac{\partial J(\theta_0, \theta_1)}{\partial \theta_0} = \frac{1}{N} \sum_{i=1}^{N} \left(h_\theta(x^{(i)}) - y^{(i)} \right) \tag{35}$$

Gradiente de la función de costo respecto a θ_1

$$\frac{\partial J(\theta_0, \theta_1)}{\partial \theta_1} = \frac{1}{N} \sum_{i=1}^{N} x^{(i)} \left(h_\theta(x^{(i)}) - y^{(i)} \right) \tag{36}$$

Optimización del Gradiente: Para minimizar $J(\theta_0, \theta_1)$, se pueden utilizar métodos iterativos como el descenso por gradiente

$$\theta_j := \theta_j - \alpha \cdot \frac{\partial J(\theta)}{\partial \theta_j}, \quad j = 0, 1 \tag{37}$$

donde α es la tasa de aprendizaje.
Solución

$$\theta_0 = \frac{1}{N} \left\{ \sum_{i=1}^{N} y^{(i)} - \theta_1 \sum_{i=1}^{N} x^{(i)} \right\} \tag{38}$$

$$\theta_1 = \frac{N \sum_{i=1}^{N} y^{(i)} x^{(i)} - \sum_{i=1}^{N} y^{(i)} \sum_{i=1}^{N} x^{(i)}}{N \sum_{i=1}^{N} (x^{(i)})^2 - (\sum_{i=1}^{N} x^{(i)})^2} \tag{39}$$

Para el caso multivariado sería

$$h_\theta\left(x\right) = \sum_{i=1}^{d} \theta_i x_i + \theta_0 = \sum_{i=0}^{d} \theta_i x_i,\ x_0 = 1 \tag{40}$$

es decir,

$$h_\theta(x) \ = \theta^T X,\ X = \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_d \end{pmatrix} \tag{41}$$

$$J(\theta) \ = J(\theta_0, \theta_1, \ldots, \theta_d) = \tfrac{1}{2N} \textstyle\sum_{i=1}^{N} \left(\theta^T x^{(i)} - y^{(i)}\right)^2. \tag{42}$$

$$h_\theta(\mathbf{X}) = \theta^T \mathbf{X} = \mathbf{X}^T \theta \tag{43}$$

$$\hat{\mathbf{y}} \ = \mathbf{X}\theta \quad \Leftrightarrow \quad \begin{bmatrix} \hat{y}^{(1)} \\ \hat{y}^{(2)} \\ \vdots \\ \hat{y}^{(N)} \end{bmatrix} = \begin{bmatrix} h_\theta \mathbf{x}^{(1)} \\ h_\theta \mathbf{x}^{(2)} \\ \vdots \\ h_\theta \mathbf{x}^{(N)} \end{bmatrix} = \begin{bmatrix} x_0^{(1)} & x_1^{(1)} & \cdots & x_d^{(1)} \\ x_0^{(2)} & x_1^{(2)} & \cdots & x_d^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ x_0^{(N)} & x_1^{(N)} & \cdots & x_d^{(N)} \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix} \tag{44}$$

donde $\mathbf{X} \in \mathbb{R}^{N\times(d+1)}$, $\hat{\mathbf{y}} \in \mathbb{R}^{N\times 1}$ y $\theta \in \mathbb{R}^{(d+1)\times 1}$. Entonces

$$\begin{aligned} J(\theta) &= \frac{1}{2N} \sum_{i=1}^{N} \left(\theta^T \mathbf{x}^{(i)} - y^{(i)}\right)^2 = \frac{1}{2N} \sum_{i=1}^{N} \left(\hat{y}^{(i)} - y^{(i)}\right)^2 \\ &= \frac{1}{2N} |\hat{\mathbf{y}} - \mathbf{y}|_2^2 = \frac{1}{2N} (\hat{\mathbf{y}} - \mathbf{y})^T (\hat{\mathbf{y}} - \mathbf{y}) = \frac{1}{2N} \left(\mathbf{X}\theta - y\right)^\top \left(\mathbf{X}\theta - y\right) \\ &= \frac{1}{2N} \left\{\theta^\top \left(\mathbf{X}^\top \mathbf{X}\right) \theta - \theta^\top \mathbf{X}^\top y - y^\top \mathbf{X}\theta + y^\top y\right\} \\ &= \frac{1}{2N} \left\{\theta^\top \left(\mathbf{X}^\top \mathbf{X}\right) \theta - \left(\mathbf{X}^\top y\right)^\top \theta - \left(\mathbf{X}^\top y\right)^\top \theta + y^\top y\right\} \\ &= \frac{1}{2N} \left\{\theta^\top \left(\mathbf{X}^\top \mathbf{X}\right) \theta - 2\left(\mathbf{X}^\top y\right)^\top \theta + y^\top y\right\} \end{aligned}$$

por lo tanto

$$J(\theta) = \frac{1}{2N} (\mathbf{X}\theta - \mathbf{y})^\top (\mathbf{X}\theta - \mathbf{y}).$$

Recordemos que $\theta^\top \mathbf{X}^\top \mathbf{y} = (\mathbf{X}^\top \mathbf{y})^\top \theta$, $(\mathbf{X}^\top \mathbf{y})^\top = \mathbf{y}^\top \mathbf{X}$ y $(\mathbf{a}^\top \mathbf{b}) = (\mathbf{b}^\top \mathbf{a})$, por lo tanto podemos reescribir:

$$J(\theta) = \frac{1}{2N}\left(\theta^\top \left(\mathbf{X}^\top \mathbf{X}\right)\theta - 2\left(\mathbf{X}^\top y\right)^\top \theta + \mathbf{y}^\top \mathbf{y}\right). \tag{45}$$

Calculando el gradiente e igualando a cero:

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= -\frac{1}{2N}\left\{\boldsymbol{\theta}^\top (X^\top X)\boldsymbol{\theta} - 2(X^\top \mathbf{y})^\top \boldsymbol{\theta} + \mathbf{y}^\top \mathbf{y}\right\} && (46)\\ &= \frac{1}{2N}\left\{2X^\top X\boldsymbol{\theta} - 2X^\top \mathbf{y}\right\}\nabla_{\boldsymbol{\theta}} && (47)\\ J(\boldsymbol{\theta}) &= 0 \Leftrightarrow X^\top X\boldsymbol{\theta} = X^\top \mathbf{y} \Leftrightarrow \boldsymbol{\theta} = (X^\top X)^{-1} X^\top \mathbf{y} && (48) \end{aligned}$$

Alternativamente (gradiente descendente)

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \theta_j} = \frac{1}{N}\sum_{i=1}^{N}\left(h_\theta(x^{(i)}) - y^{(i)}\right) x_j^{(i)} \tag{49}$$

Nota 21.
- *Resolver por gradiente descentiente, y calcular $(X^\top X)^{-1}$ puede ser difícil.*
- $(X^\top X)^\top = X^\top (X^\top)^\top = X^\top X$.

Ejemplo 67.1. *Ahora calculemos la log-verosimilitud considerando que cada variable se distribuye Bernoulli: $p\,(x|\theta) = \theta\; x\,(1-\theta)^{1-x}$ para $x = 0, 1$. Sea $x_1, x_2, \ldots, x_n, \quad \theta = \frac{1}{n}\sum_{i=1}^{n} x_i$, entonces $\theta = \sum_{i=1}^{n} \frac{x_i}{n}$ es el valor más probable para estimar θ. La función de verosimilitud conjunta*

$$\Psi(x_1, x_2, \ldots, x_n; \theta) = \prod_{i=1}^{n} f(x_i; \theta) = \theta^{\sum_{i=1}^{n} x_i}(1-\theta)^{n - \sum_{i=1}^{n} x_i}, \tag{50}$$

entonces la función de Log-verosimilitud está dada por:

$$\log L(\theta) = \sum_{i=1}^{n} x_i \log\theta + (n - \sum_{i=1}^{n} x_i)\log(1-\theta), \tag{51}$$

calculando la derivada de la log-verosimilitud

$$\begin{aligned} \frac{\partial \log L(\theta)}{\partial \theta} &= \frac{\sum_{i=1}^{n} x_i}{\theta} - \frac{n - \sum_{i=1}^{n} x_i}{1-\theta} && (52)\\ &= \frac{(1-\theta)\sum_{i=1}^{n} x_i - (n - \sum_{i=1}^{n} x_i)\theta}{\theta(1-\theta)} = \frac{\sum_{i=1}^{n} x_i - n\theta}{\theta(1-\theta)}, && (53) \end{aligned}$$

igualando a cero y resolviendo

$$\frac{\sum_{i=1}^{n} x_i - n\theta}{\theta(1-\theta)} = 0 \Leftrightarrow \theta = \frac{\sum_{i=1}^{n} x_i}{n}. \tag{54}$$

Ejemplo 67.2. *Supongamos que se tienen* $\mathcal{D} = \left\{u^{(1)}, u^{(2)}, \ldots, u^{(N)}\right\}$ *observaciones, supongamos además que se tienen datos generados con distrubución* $U \sim (U; \theta)$*. Calculemos la función de verosimlitud.*

$$\mathcal{L}(\theta) = \prod_{i=1}^{N} p\left(u^{(i)}; \theta\right) \tag{55}$$

donde

$$\theta_{ML} = \arg\max_{\theta} \mathcal{L}(\theta) = \arg\max_{\theta} \sum_{i=1}^{N} \log p\left(u^{(i)}; \theta\right) \tag{56}$$

donde tanto $log(f(x))$ *y* $\arg\max_\theta$ *son funciones monótonas crecientes. supongamos que se tiene un ruido gaussiano com media* 0 *y varianza* σ^2*, entonces*

$$y^{(i)} = h_\theta\left(x^{(i)}\right) + \epsilon^{(i)} = \theta^\top \mathbf{X}^{(i)} + \epsilon^{(i)}, \tag{57}$$

por lo tanto

$$y^{(i)} \sim N\left(\theta^\top \mathbf{X}^{(i)}, \sigma^2\right), \tag{58}$$

entonces

$$\begin{aligned} p\left(y|\mathbf{X}, \theta, \sigma^2\right) &= \prod_{i=1}^{N} p\left(y|\mathbf{x}^{(i)}, \theta, \sigma^2\right) = \prod_{i=1}^{N} \left(2\pi\sigma^2\right) e^{-\frac{1}{2\sigma^2}\left(y^{(i)} - \theta^\top \mathbf{x}^{(i)}\right)^2} && (59)\\ &= \left(2\pi\sigma^2\right)^{-\frac{N}{2}} e^{-\frac{1}{2\sigma^2}\sum_{i=1}^{2}\left(y^{(i)} - \theta^\top \mathbf{x}^{(i)}\right)^2} && (60)\\ &= \left(2\pi\sigma^2\right)^{-\frac{N}{2}} e^{-\frac{1}{2\sigma^2}(y - \mathbf{X}\theta)^\top (y - \mathbf{X}\theta)} && (61) \end{aligned}$$

entonces la verosimilitud es

$$p\left(y|\mathbf{X}, \theta, \sigma^2\right) = \left(2\pi\sigma^2\right)^{-\frac{N}{2}} e^{-\frac{1}{2\sigma^2}(y - \mathbf{X}\theta)^\top (y - \mathbf{X}\theta)} \tag{62}$$

y la log-verosimilitud es

$$\mathcal{L}\left(\theta, \sigma^2\right) = -\frac{N}{2}\log\left(2\pi\sigma^2\right)\left[-\frac{1}{2\sigma^2}\left(y - \mathbf{X}\theta\right)^\top \left(y - \mathbf{X}\theta\right)\right]. \tag{63}$$

Maximizar la log-verosimilitud con respecto aθ *es equivalente a maximizar* $-\left(y - \mathbf{X}\theta\right)^\top \left(y - \mathbf{X}\theta\right)$ *que a su vez es equivalente a minimizar* $\left(y - \mathbf{X}\theta\right)^\top \left(y - \mathbf{X}\theta\right)$*.*

68. Tema nuevo

Se define la función sigmoide

$$\sigma(u) = \frac{1}{1+e^{-u}} \quad \Rightarrow \text{logistic regression classifier} \tag{64}$$

donde la regla de decisión para y

$$y = \sigma(h_{\boldsymbol{\theta}}(x)) = \sigma(\boldsymbol{\theta}^\top x) \tag{65}$$

Matemáticamente, la probabilidad de que un ejemplo pertenezca a la clase 1 es:

$$p\left(y^{(i)} = 1 \mid x^{(i)}; \boldsymbol{\theta}\right) = \sigma(\boldsymbol{\theta}^\top x^{(i)}) \tag{66}$$

$$p\left(y^{(i)} = 0 \mid x^{(i)}; \boldsymbol{\theta}\right) = 1 - \sigma(\boldsymbol{\theta}^\top x^{(i)}) \tag{67}$$

la probabilidad conjunta en función de $y^{(i)}$

$$p\left(y^{(i)} \mid x^{(i)}; \boldsymbol{\theta}\right) = \sigma(\boldsymbol{\theta}^\top x^{(i)})^{y^{(i)}} \cdot \left[1 - \sigma(\boldsymbol{\theta}^\top x^{(i)})\right]^{1-y^{(i)}} \tag{68}$$

mientras que la probabilidad conjunta de todas las etiquetas

$$\prod_{i=1}^{N} \sigma(\boldsymbol{\theta}^\top x^{(i)})^{y^{(i)}} \left(1 - \sigma(\boldsymbol{\theta}^\top x^{(i)})\right)^{(1-y^{(i)})} \tag{69}$$

La log-verosimilitud para regresión logística está dada por:

$$\ell(\boldsymbol{\theta}) = \sum_{i=1}^{N} y^{(i)} \log\left(\sigma(\boldsymbol{\theta}^\top x^{(i)})\right) + (1 - y^{(i)}) \log\left(1 - \sigma(\boldsymbol{\theta}^\top x^{(i)})\right) \tag{70}$$

Antes de calcular la derivada, recordemos:

$$\sigma(z) = \frac{1}{1+e^{-z}} \tag{71}$$

con derivada

$$\frac{d}{dz}\sigma(z) = \frac{d}{dz}\left(1+e^{-z}\right)^{-1} = -(1+e^{-z})^{-2} \cdot (-e^{-z}) = \frac{e^{-z}}{(1+e^{-z})^2} \tag{72}$$

además:

$$\sigma(z) = \frac{1}{1+e^{-z}} \Rightarrow 1 - \sigma(z) = \frac{e^{-z}}{1+e^{-z}} \tag{73}$$

$$\Rightarrow \sigma(z)(1-\sigma(z)) = \frac{e^{-z}}{(1+e^{-z})^2} \tag{74}$$

$$\therefore \frac{d}{dz}\sigma(z) = \sigma(z)(1-\sigma(z)) \tag{75}$$

Derivando la log-verosimilitud respecto a $theta_j$ la función $\ell(\boldsymbol{\theta})$:

$$\frac{\partial \ell(\boldsymbol{\theta})}{\partial \theta_j} = \sum_{i=1}^{N} \left[y^{(i)} \log \sigma(\boldsymbol{\theta}^\top x^{(i)}) + (1 - y^{(i)}) \log(1 - \sigma(\boldsymbol{\theta}^\top x^{(i)})) \right] \tag{76}$$

$$= \sum_{i=1}^{N} \left[\frac{y^{(i)}}{\sigma(\boldsymbol{\theta}^\top x^{(i)})} - \frac{1 - y^{(i)}}{1 - \sigma(\boldsymbol{\theta}^\top x^{(i)})} \right] \cdot \frac{d}{d\theta_j} \sigma(\boldsymbol{\theta}^\top x^{(i)}) \tag{77}$$

$$= \sum_{i=1}^{N} \left[\frac{y^{(i)}}{\sigma(\boldsymbol{\theta}^\top x^{(i)})} - \frac{1 - y^{(i)}}{1 - \sigma(\boldsymbol{\theta}^\top x^{(i)})} \right] \cdot \sigma(\boldsymbol{\theta}^\top x^{(i)}) \sigma(1 - \boldsymbol{\theta}^\top x^{(i)}) x_j^{(i)} \tag{78}$$

$$= \sum_{i=1}^{N} \left[\frac{y^{(i)} - \sigma(\boldsymbol{\theta}^\top x^{(i)})}{\sigma(\boldsymbol{\theta}^\top x^{(i)})(1 - \sigma(\boldsymbol{\theta}^\top x^{(i)}))} \right] \cdot \sigma(\boldsymbol{\theta}^\top x^{(i)}) \sigma(1 - \boldsymbol{\theta}^\top x^{(i)}) x_j^{(i)} \tag{79}$$

$$= \sum_{i=1}^{N} \left[y^{(i)} - \sigma(\boldsymbol{\theta}^\top x^{(i)}) \right] x_j^{(i)} \tag{80}$$

la cual es la función recursiva para calcular el gradiente.

Resumen traducido

Las técnicas de regresión son versátiles en su aplicación a la investigación médica porque pueden medir asociaciones, predecir resultados y controlar los efectos de variables de confusión. Como una de estas técnicas, la regresión logística es una forma eficiente y poderosa de analizar el efecto de un grupo de variables independientes sobre un resultado binario cuantificando la contribución única de cada variable independiente. Utilizando componentes de regresión lineal reflejados en la escala logit, la regresión logística identifica iterativamente la combinación lineal más fuerte de variables con la mayor probabilidad de detectar el resultado observado.

Las consideraciones importantes al realizar una regresión logística incluyen la selección de variables independientes, asegurando que se cumplan los supuestos relevantes y eligiendo una estrategia adecuada de construcción del modelo. Para la selección de variables independientes, uno debe guiarse por factores como teoría aceptada, investigaciones empíricas previas, consideraciones clínicas y análisis estadísticos univariados, reconociendo las posibles variables de confusión que deben ser consideradas.

Los supuestos básicos que deben cumplirse para la regresión logística incluyen independencia de errores, linealidad en el logit para variables continuas, ausencia de multicolinealidad y falta de valores atípicos fuertemente influyentes. Adicionalmente, debe haber un número adecuado de eventos por variable independiente para evitar un modelo sobreajustado, con un mínimo comúnmente recomendado de "reglas prácticas" que van de 10 a 20 eventos por covariable.

Respecto a las estrategias de construcción de modelos, los tres tipos generales son: directa/estándar, secuencial/jerárquica y por pasos/estadística, cada uno con un énfasis y propósito diferente. Antes de llegar a conclusiones definitivas a partir de los resultados de cualquiera de estos métodos, se debe cuantificar formalmente la validez interna del modelo

(es decir, su replicabilidad dentro del mismo conjunto de datos) y su validez externa (es decir, su generalizabilidad más allá de la muestra actual).

El ajuste general del modelo de regresión logística a los datos de muestra se evalúa utilizando varias medidas de bondad de ajuste, donde un mejor ajuste se caracteriza por una menor diferencia entre los valores observados y los valores predichos por el modelo. También se recomienda el uso de estadísticas de diagnóstico para evaluar aún más la adecuación del modelo. Finalmente, los resultados para las variables independientes suelen reportarse como razones de momios (odds ratios, ORs) con intervalos de confianza (IC) del 95 %.

Resumen

Este artículo presenta una revisión exhaustiva de las aplicaciones del aprendizaje automático (Machine Learning, ML) dentro del campo de la inteligencia artificial (IA). Se examinan los principales tipos de aprendizaje: supervisado, no supervisado, por refuerzo y sistemas de recomendación, destacando sus aplicaciones prácticas y proponiendo ideas futuras como el "doctor virtualz la "máquina del tiempo informativa".

69. Introducción

Un agente inteligente en IA interactúa con el entorno mediante sensores y actuadores. Su inteligencia depende de la política de control que traduce entradas en acciones. ML permite alcanzar inteligencia humana simulada sin programación explícita. Aplicaciones incluyen búsqueda web, reconocimiento de fotos, y filtros de spam. Se destaca su uso en robótica autónoma, biología computacional y Big Data.

70. Aprendizaje Automático

Se definen conceptos clave del aprendizaje automático:

- Arthur Samuel lo define como la capacidad de una computadora para aprender sin ser programada.
- Tom Mitchell propone una definición formal basada en experiencia (E), tarea (T) y medida de rendimiento (P). Si el desempeño en T, medido a trav¬'es de P, mejora con la experiencia E, entonces el programa es llamando un programa de Machine Learning.

Se destaca el ejemplo del programa de damas de Samuel, que mejora jugando contra sí mismo.

71. Tipos de Algoritmos de Aprendizaje

71.1. Aprendizaje Supervisado

Entrenamiento con datos etiquetados, comparando salida esperada con salida computada. Ejemplo: estimación del precio de viviendas.

71.2. Aprendizaje No Supervisado

Unsupervised learning is termed as learned by its own by discovering and adopting, based on the input pattern. In this learning the data are divided into different clusters and hence the learning is called a clustering algorithm Descubre patrones ocultos sin datos etiquetados. Agrupa datos en clústeres, como en Google News.

71.3. Aprendizaje por Refuerzo

Aprende mediante recompensas por buenas acciones y penalizaciones por errores, sin ejemplos explícitos. Se otorga una recompensa por una salida correcta y una penalización por una salida incorrecta. El aprendizaje por refuerzo se diferencia del aprendizaje supervisado en que nunca se presentan pares de entrada/salida correctos, ni se corrigen explícitamente las acciones subóptimas

71.4. Sistemas de Recomendación

Personalizan contenido para usuarios mediante recomendaciones basadas en contenido o colaborativas. Usado en sitios de comercio electrónico. There are mainly two approaches: content based recommendation and collaborative recommendation, which help the user for obtaining and mining data, making intelligent and novel recommendations, ethics.

72. Aplicaciones del Aprendizaje Automático

72.1. Aprendizaje No Supervisado

- **Clasificación de ADN:** Agrupamiento de individuos por genes usando microarrays.
- **Clústeres de computadores:** Organiza centros de datos eficientemente.
- **Redes sociales:** Detecta amistades, grupos, y patrones de comunicación.
- **Segmentación de mercado:** Descubre segmentos automáticamente a partir de datos de clientes.
- **Datos astronómicos:** Analiza formación de galaxias y detecta anomalías (objetos o patrones extraños).

- **Problema del cóctel:** Separa fuentes de audio combinadas usando algoritmos no supervisados.
- **Registros médicos y biología computacional:** Mejora diagnóstico, comprensión genómica y clasificación de cáncer.
- **Detección de actividad de voz (SAD):** Identifica momentos de habla versus silencio.
- **Verificación de hablantes:** Usa análisis acústico para autenticación.

72.2. Aprendizaje Supervisado

- **Correo electrónico:** Respuestas automáticas, organización de carpetas, resumen de hilos, y filtro de spam.
- **Reconocimiento de escritura:** Identifica direcciones en sobres.
- **Reconocimiento facial y de voz:** Aplicado en seguridad y redes sociales.
- **Recuperación de información:** Búsqueda eficiente y personalizada.
- **Sistemas operativos:** Predicen apps frecuentes para carga rápida.
- **Detección de intrusos y anómalias:** Usa secuencias de acciones para detectar comportamientos anormales.
- **Clasificación de textos:** Asigna documentos a categorías temáticas.
- **Optimización de centros de datos:** Usa redes neuronales para eficiencia energética.
- **Radio cognitiva:** Mejora procesamiento de señales mediante reducción de dimensionalidad y SVM.
- **Finanzas computacionales:** Predice movimientos del mercado bursátil.
- **Interfaces cerebro-máquina (BCI):** Permite controlar dispositivos con actividad cerebral.
- **Producción musical:** Clasifica géneros, transcribe, detecta ritmo e instrumentos.

72.3. Sistemas de Recomendación

- **Aprendizaje móvil:** Ofrece contenido educativo personalizado.
- **Publicidad computacional:** Asocia usuarios con anuncios en contexto.
- **Análisis de sentimientos:** Clasifica opiniones como positivas o negativas.
- **Minería de bases de datos:** Extrae patrones de grandes volúmenes de datos.
- **Programas auto-personalizables:** Aprenden preferencias del usuario y adaptan interfaces.

72.4. Aprendizaje por Refuerzo

- **Predicción de tráfico:** Sistemas que estiman condiciones futuras de tráfico.
- **Juegos de computadora:** IA que mejora experiencia de juego.
- **Maquinaria autónoma:** Aprenden tareas como volar helicópteros.
- **Análisis bursátil:** Usa SVM y refuerzo para tomar decisiones financieras.
- **Ambientes de aprendizaje ubicuo:** Simulaciones realistas para evaluar habilidades clínicas.

73. Impresiones y Perspectivas

Se observa que la capacidad de análisis de datos masivos ha impulsado el desarrollo de agentes autónomos. Se destaca el papel del aprendizaje continuo y la necesidad de conjuntos de datos actualizados. Entre las propuestas futuras se incluyen la máquina del tiempo informativa y el doctor virtual.

74. Conclusión

El aprendizaje automático ha demostrado ser esencial para la automatización inteligente, con aplicaciones en múltiples disciplinas. Aunque hay limitaciones en la calidad y disponibilidad de los datos, el campo sigue creciendo. Se destaca la necesidad de mejorar continuamente los algoritmos y entrenarlos con datos diversos y actualizados.