Week #8: Stochastic Inference

March 10, 2025

Definition. Program

A **program** P is defined as a tuple (M, θ), where:

- $M : I \to O$ is a function from the input space I to the output space O. M encapsulates the operational logic of the program, defining specific computations to transform each input $i \in I$ into an output $o \in O$.
- $\theta \in \Theta$ are the parameters of the model M, with Θ representing the parameter space. These parameters adjust the function M, allowing it to be specialized or generalized according to different tasks or datasets.

In operational terms, for a given input $x \in I$, the program applies M parameterized by θ to produce an output y, expressed as $y = M_\theta(x)$. This computation models the essential mechanism by which the program processes data and generates results.

[Sorting Program] Let $I = \{x \mid x \text{ is a list of integers}\}$ and $O = \{y \mid y \text{ is a list of integers sorted in non-decre}$
Define a sorting program with $M : I \to O$ such that for any $x \in I$, $M(x) = y$, where y is the permutation of x sorted in non-decreasing order.

Another example involves image recognition tasks. Let's consider an image as a matrix:

[Dog/Cat Identification Program] Consider Im as a matrix representing a grayscale image, where each element Im_{ij} denotes the pixel intensity at position (i, j). Let $I = \{\text{Im} \mid \text{Im is a } m \times n \text{ matrix}\}$ and $O = \{'dog', 'cat'\}$. Define a classification program with $M : I \to O$ trained to map an input image $\text{Im} \in I$ to a label $y \in O$ based on its content.

These examples demonstrate how a program as defined by the function M and its parameters θ, effectively maps inputs to outputs within specified computational domains.

1 Linear Regression

Linear regression is a fundamental statistical method for prediction, modeling the relationship between a dependent variable and one or more explanatory variables.

Consider a dataset $\{(u^{(i)}, v^{(i)})\}_{i=1}^N$, with each $u^{(i)} \in \mathbb{R}^p$ as a vector of p input features, and $v^{(i)} \in \mathbb{R}$ as the corresponding observed output. We define the model function M as:

$$M_\theta(u^{(i)}) = \theta_0 + \sum_{j=1}^{p} \theta_j u_j^{(i)},$$

where $\theta = (\theta_0, \theta_1, \ldots, \theta_p)$ are the model parameters, including the intercept θ_0 and coefficients θ_j for each feature.

The goal of linear regression is to find the parameter set θ that minimizes the sum of squared residuals, effectively fitting the model to the data. This objective is formulated as:

$$\min \sum^{N} \left(v^{(i)} - M_\theta(u^{(i)})\right)^2 .$$

where v is the vector of observed outputs.

So, given data, we can 'train' a model directly using equation 1 and obtain a program, to predict the outcome of future input features.

2 Logistic Regression

Logistic regression is a key method for binary classification, predicting probabilities of binary outcomes effectively. This approach is particularly suitable for scenarios where the response variable is categorical, such as in medical diagnostics, spam detection, or credit scoring.

Consider a dataset $\{(u^{(i)}, v^{(i)})\}_{i=1}^{N}$, where each $u^{(i)} \in \mathbb{R}^p$ is a vector of p input features and $v^{(i)} \in \{0, 1\}$ denotes the binary outcomes. The logistic regression model function M is defined as:

$$M_\theta(u^{(i)}) = \sigma(\theta^T u^{(i)}),$$

where

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

is the logistic sigmoid function that converts a linear combination of inputs into a probability.

Probability Model and Loss Function: The outcome $v^{(i)}$ is modeled as a Bernoulli random variable with the success probability $p^{(i)} = M_\theta(u^{(i)})$. The probability of observing $v^{(i)}$ given $u^{(i)}$ under model parameters θ is given by:

$$p(v^{(i)} \mid u^{(i)}, \theta) = (p^{(i)})^{v^{(i)}} (1 - p^{(i)})^{1 - v^{(i)}}.$$

The likelihood function $L(\theta)$, representing the joint probability of observing all the $v^{(i)}$ given $u^{(i)}$ for $i = 1, \ldots, N$, is:

$$L(\theta) = \prod_{i=1}^{N} (p^{(i)})^{v^{(i)}} (1 - p^{(i)})^{1 - v^{(i)}},$$

and the log likelihood, which we seek to maximize, is:

$$\ell(\theta) = \sum_{i=1}^{N} \left[v^{(i)} \log(p^{(i)}) + (1 - v^{(i)}) \log(1 - p^{(i)}) \right].$$

The goal is to find θ that maximizes $\ell(\theta)$, or equivalently minimizes the negative of $\ell(\theta)$:

$$\min_\theta -\ell(\theta) = -\sum_{i=1}^{N} \left[v^{(i)} \log(p^{(i)}) + (1 - v^{(i)}) \log(1 - p^{(i)}) \right].$$

Optimization and Model Training: This optimization problem is generally solved using iterative techniques such as gradient descent, where each step aims to adjust θ to reduce the negative log likelihood, improving the model fit to the data.

Upon training, the logistic regression model M can then be used as a predictive tool, estimating the probability that new inputs fall into one of the binary categories. This capability makes logistic regression a powerful class of programs within the framework of stochastic modeling for prediction.

[Predicting Programming Project Success] Logistic regression is employed to predict the success of programming projects based on evaluations from two experts. Each expert assesses the projects independently, and the logistic model predicts the probability of project success using these assessments.

Given several programming projects evaluated by the experts, the logistic regression model is formulated as:

$$p(\text{Success}) = \frac{1}{1+e^{-z}},$$

where z is a linear combination of the expert ratings:

$$z = \beta_0 + \beta_1 \text{Expert}_1 + \beta_2 \text{Expert}_2.$$

Model Training: The model is trained using a dataset where each project is labeled as '1' (success) or '0' (failure) based on historical outcomes. The coefficients β_1 and β_2 are optimized to minimize the prediction error, using methods like gradient descent.

Prediction: The success probability $p(\text{Success})$ for each project is calculated. Projects with $p(\text{Success}) \geq 0.5$ are predicted as likely to succeed.

Decision-making:

- Projects predicted to succeed may receive additional resources or prioritization.
- Projects with low success probabilities might be reviewed or modified.

Table 1: Expert Ratings and Project Outcomes

Project ID	**Expert 1**	**Expert 2**	**Outcome (Success=1)**
Project A	8	5	1
Project B	6	5	0
...	...	...	...
Project N	3	9	?

3 Support Vector Machines

Support Vector Machines (SVMs) are powerful supervised learning models used for classification and regression tasks. The primary objective in SVM is to establish a hyperplane that maximally separates the classes in the feature space.

Definition and Composition of SVM

Definition. Support Vector Machine

A **Support Vector Machine** M is characterized by a separating hyperplane which divides the feature space I into classes that maximize the margin between the nearest members of different classes, which are termed as support vectors.

Mathematical Model

The optimal hyperplane can be represented by the equation:

$$\mathbf{w}^T\mathbf{x} + b = 0,$$

where $\mathbf{w}$ is the weight vector perpendicular to the hyperplane, $\mathbf{x}$ is the input feature vector, and b is the bias.

The objective is to maximize the margin between the hyperplane and the nearest training data point of any class, which is mathematically given by:

$$\max_{\mathbf{w},b} \frac{2}{\|\mathbf{w}\|},$$

subject to the constraints for each data point $(\mathbf{x}_i, y_i)$ in the training set:

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1, \quad \forall i.$$

Kernel Trick

For non-linear classification, SVMs use the kernel trick to transform the input space into a higher-dimensional space where a linear separator might exist. The kernel function K relates to the dot product of two feature vectors in this new space:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j),$$

where ϕ is the transformation function to the higher-dimensional space.

Optimization and Learning

The optimization problem in SVMs involves minimizing a quadratic function subject to linear constraints, which is typically solved using methods such as Sequential Minimal Optimization (SMO). Once the model parameters $\mathbf{w}$ and b are determined, the SVM model M acts as a program that assigns new inputs $\mathbf{x}$ to one of the categories based on the sign of $\mathbf{w}^T \mathbf{x} + b$.

Implementation in Data Analysis

In practice, SVMs are used to solve a variety of real-world problems such as image classification, voice recognition, and biological classification, demonstrating robust performance across a wide range of data types and complexities.

This structured approach allows SVMs to model complex relationships in data by maximizing the margin between class boundaries, providing a strong predictive performance characterized by generalization to unseen data.

Generalized Additive Models

Definition. Generalized Linear Models

Generalized Linear Models (GLMs) are a broad class of models that extend traditional linear regression by allowing for a response variable $v^{(i)}$ whose distribution is a member of the exponential family, and by utilizing a link function to relate the expected value of the response to the linear predictor. This flexibility allows GLMs to handle a wider range of data types than standard linear regression.

Consider a dataset $\{(u^{(i)}, v^{(i)})\}_{i=1}^N$, with each $u^{(i)} \in \mathbb{R}^p$ representing a vector of p input features, and $v^{(i)}$ representing the observed responses. The systematic component of a GLM, or the linear predictor, is defined as:

$$\eta^{(i)} = \theta^T u^{(i)},$$

where $\eta^{(i)}$ functions as a combination of the input features weighted by the model parameters θ. The link between the mean of the response distribution $\mu^{(i)} = E(v^{(i)})$ and the linear predictor is established through a link function g:

$$\mu^{(i)} = g^{-1}(\eta^{(i)}).$$

Link Function and Response Distribution: The choice of the link function, such as the logit for the binomial distribution or the natural log for the Poisson distribution, is crucial and depends on the nature of the response variable. This ensures the appropriateness of the model to the data and allows for the modeling of variables that are counts, proportions, or times until an event.

Loss Function and Estimation: The estimation of parameters in GLMs is generally performed through maximum likelihood estimation (MLE). The likelihood function, based on the chosen exponential family distribution, is:

$$L(\theta) = \prod_{i=1}^{N} f(v^{(i)} \mid u^{(i)}; \theta),$$

where f is the probability density or mass function. The corresponding loss function, utilized during the optimization process, is the negative log-likelihood:

$$\min_{\theta} -\log L(\theta) = -\sum_{i=1}^{N} \log f(v^{(i)} \mid u^{(i)}; \theta).$$

Generalized Additive Models (GAMs): As a subclass of GLMs, Generalized Additive Models (GAMs) further generalize the linear predictor to include non-linear terms via smooth functions. In GAMs, the linear predictor may include terms like:

$$\eta^{(i)} = \beta_0 + s_1(u_1^{(i)}) + s_2(u_2^{(i)}) + \cdots + s_p(u_p^{(i)}),$$

where s_j are smooth functions of the predictors that capture non-linear relationships. This addition enhances the model's ability to capture complex patterns in the data, making GAMs particularly useful in ecological and environmental modeling.

Optimization and Practical Implementation: Optimization methods such as Newton-Raphson or iterative re-weighted least squares (IRLS) are commonly employed to find the best-fitting model parameters θ that minimize the loss function. This process is key to ensuring that the GLM or GAM accurately reflects the underlying data relationships.

Once trained, both GLM and GAM function as predictive programs, mapping input features to predicted outcomes effectively. This capability makes them invaluable tools for diverse applications in predictive modeling across numerous scientific disciplines.

4 Neural Networks

Neural networks are sophisticated mathematical frameworks designed to emulate certain processing patterns found in biological neural systems. Below, we describe the foundational components and the mathematical structure of neural networks.

General Structure

Neural networks consist of layers of interconnected nodes called neurons. Each neuron in a layer is connected to several other neurons in the previous and next layers, forming a network that can propagate data forward from input to output.

Definition. Neural Network Architecture

A Neural Network M is formally defined by its architecture, which includes layers of neurons, and a set of weights $\mathbf{w}$. The architecture determines how inputs are transformed through the network:

$$M(\mathbf{x};\mathbf{w}) = f_L \circ \cdots \circ f_1(\mathbf{x};\mathbf{w}_1,\ldots,\mathbf{w}_L)$$

where $\mathbf{x}$ is the input vector to the network, $\mathbf{w}_i$ denotes the weights for the i-th layer, and f_i are the transformation functions specific to each layer.

Key Operations

Linear Combination: Each neuron computes a weighted sum of its inputs, which serves as the input to a non-linear activation function. The output of each neuron is given by:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right),$$

where x_i are the inputs to the neuron, w_i are the corresponding weights, b is a bias term, and f is a non-linear activation function.

Convolutional Layers (specific to CNNs): In CNNs, convolutional layers apply filters to the input, capturing spatial features:

$$(S * K)(i,j) = \sum_{u=1}^{m}\sum_{v=1}^{m} S(i+u, j+v)K(u,v),$$

where S is the input matrix, K is a kernel of size $m \times m$, and i, j index the output feature map.

Activation Functions: Activation functions such as ReLU (Rectified Linear Unit) introduce non-linear properties to the network:

$$f(x) = \max(0, x),$$

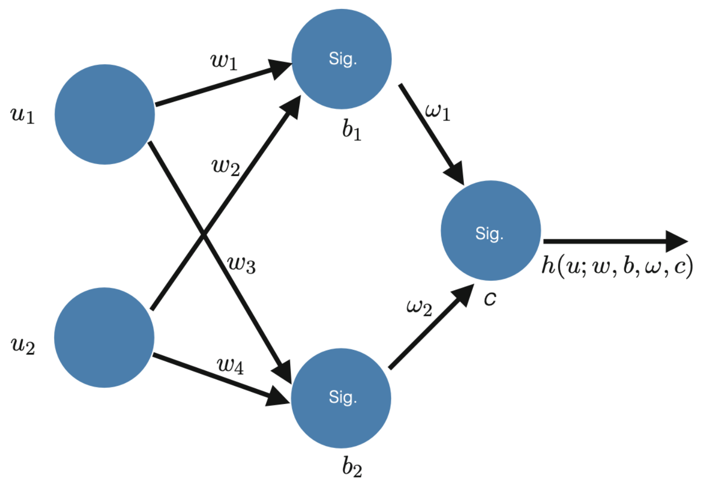
allowing the network to learn complex patterns in data.

Learning and Optimization

Backpropagation: Neural networks adjust their weights through backpropagation, where errors between the predicted and actual outputs are propagated back through the network to update the weights.

Gradient Descent: This optimization method updates weights by moving in the direction that minimally increases the loss function:

$$\mathbf{w}_{new} = \mathbf{w}_{old} - \eta \nabla \mathcal{L}(\mathbf{w}),$$



A simple neural network

where η is the learning rate and $\nabla\mathcal{L}(\mathbf{w})$ is the gradient of the loss function with respect to the weights.

We will see in detail the main stochastic optimization methods in next lecture.