# Mathematical Formulas of Major Machine Learning Models

December 1, 2024

## 1 Introduction

Here are the fundamental mathematical formulas for major neural networks and machine learning models: Convolutional Neural Networks (CNN), Random Forests, LightGBM, Transformers (Attention Mechanism), and Support Vector Machines (SVM). Subsequently, we propose a simplified and optimized formulation integrating key aspects of each model.

## 2 Mathematical Formulas of the Models

### 2.1 Convolutional Neural Networks (CNN)

#### 2.1.1 Convolution Operation

$$\mathbf{O}(i,j,k) = \sum_{m=1}^{M} \sum_{n=1}^{N} \sum_{c=1}^{C} \mathbf{I}(i+m, j+n, c) \cdot \mathbf{F}(m,n,c,k) + \mathbf{b}(k)$$

- $\mathbf{I}$: Input (image or data volume)
- $\mathbf{F}$: Filter (network weights)
- $\mathbf{b}$: Bias
- $\mathbf{O}$: Output (feature map)
- $i, j$: Spatial position indices
- $k$: Filter index
- $M, N$: Filter dimensions
- $C$: Number of input channels

#### 2.1.2 Pooling (Max-Pooling)

$$\mathbf{P}(i,j,k) = \max\{\mathbf{O}(i+m, j+n, k) \mid m,n \in \text{pooling window}\}$$

### 2.2 Random Forests

#### 2.2.1 Prediction of a Forest of $T$ Trees

$$\hat{y} = \frac{1}{T} \sum_{t=1}^{T} \hat{y}_t$$

- $\hat{y}_t$: Prediction of tree $t$

### 2.3 LightGBM (Gradient Boosting)

#### 2.3.1 Loss Function Minimization at Each Iteration $t$

$$\mathcal{L} = \sum_{i=1}^{N} \ell(y_i, \hat{y}_i) + \Omega(f_t)$$

- $\ell$: Loss function
- $\Omega$: Regularization function
- $f_t$: New tree added

#### 2.3.2 Optimization of Gradients and Hessians

$$f_t = \arg\min_f \sum_{i=1}^{N} \left[ g_i f(\mathbf{x}_i) + \frac{1}{2} h_i f(\mathbf{x}_i)^2 \right] + \Omega(f)$$

- $g_i$: Gradient of the loss with respect to $\hat{y}_i$
- $h_i$: Hessian of the loss with respect to $\hat{y}_i$

### 2.4 Transformers (Attention Mechanism)

#### 2.4.1 Self-Attention

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V$$

- $Q$: Query vectors
- $K$: Key vectors
- $V$: Value vectors
- $d_k$: Dimension of the keys

### 2.5 Support Vector Machines (SVM)

#### 2.5.1 Optimization Problem

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 \quad \text{subject to} \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \quad \forall i$$

- $\mathbf{w}$: Weight vector
- $b$: Bias
- $y_i$: Class label
- $\mathbf{x}_i$: Input vector

## 3 Simplification and Consolidation of Formulas

To consolidate these formulas, we can identify common components and abstract certain operations. Here is a simplified formulation that integrates the key aspects of each model:

$$\text{Output} = \sigma\left(\sum_{i=1}^{N} \alpha_i \cdot f_i(\mathbf{X}; \theta_i)\right) + b$$

- $\mathbf{X}$: Input (image, feature vector, etc.)
- $f_i$: Different functions or sub-models (convolution, decision trees, attention, etc.)
- $\alpha_i$: Importance weights for each sub-model

- $\theta_i$: Parameters of the sub-models
- $\sigma$: Non-linear activation function (ReLU, Softmax, etc.)
- $b$: Bias

# 4 Optimization of the Formulation

To optimize this formulation to reduce training time and computational power while maintaining high performance, here are some strategies:

## 4.1 Compression and Pruning

- **Pruning**: Remove less important connections or neurons in neural networks to reduce model size.
- **Quantization**: Use low-precision weight representations (e.g., 8-bit instead of 32-bit).

## 4.2 Knowledge Distillation

Transfer knowledge from a large, complex model to a smaller, faster model:

$$\text{Teacher} \rightarrow \text{Student}$$

Where the "Student" model learns to mimic the outputs of the "Teacher" model.

## 4.3 Use of Efficient Architectures

Choose architectures optimized for speed and efficiency:

- **MobileNet** or **EfficientNet** for CNNs.
- **LightGBM** for fast gradient boosting models.

## 4.4 Optimized Ensemble Techniques

Combine models efficiently:

- **Bagging** for Random Forests.
- **Boosting** for LightGBM.
- **Light Ensembles** combining several simple models instead of complex ones.

## 4.5 Hyperparameter Optimization

Use techniques like Bayesian optimization or genetic algorithms to quickly find the best hyperparameters.

## 4.6 Hardware and Software Acceleration

- **GPU/TPU**: Utilize specialized processing units to speed up computations.
- **Optimized Libraries**: Use libraries like TensorFlow Lite, ONNX, or others optimized for fast execution.

# 5 Generalized Optimized Formula

By integrating the mentioned optimizations, we can envision an optimized formulation as follows:

$$\text{Output} = \sigma\left(\sum_{i=1}^{N} \alpha_i \cdot \text{Pruning}(f_i(\mathbf{X}; \theta_i))\right) + \text{Quantization}(b)$$

- **Pruning** and **Quantization** are applied to the sub-models to reduce complexity and accelerate computations.
- $\alpha_i$ are adjusted to balance the importance of different sub-models while minimizing resource requirements.

# 6 Conclusion

Although the mentioned models (CNN, Random Forests, LightGBM, Transformers, SVM) have different mechanisms and objectives, it is possible to consolidate them into a unified formulation using a modular approach. By applying advanced optimization techniques such as model compression, knowledge distillation, and the use of efficient architectures, you can create a hybrid model that performs well while requiring less training time and computational power.

However, the practical implementation of this model will require careful design and extensive experimentation to balance the different components and optimize overall performance.

If you wish to delve deeper into a specific aspect or have additional questions, feel free to let me know!