Topic 4: Deep Learning (Part I)

Tian Xie[†]

[†]**Shanghai University of Finance and Economics**

November 1, 2023 @ **Techno-Economics Research Alliance (TERA)**

A Nibble of History

- Deep learning has its theoretical foundation dating back to the 1940s.
- The field has undergone various name changes throughout its history, including "cybernetics" in the 1940s-1960s and "connectionism" in the 1980s-1990s.
- The term "deep learning" gained prominence in 2006, marking a resurgence in the field.
- Deep learning is sometimes confused with artificial neural networks (ANN), as early algorithms aimed to mimic biological learning processes.
- The modern concept of deep learning extends beyond a neuro-scientific perspective, involving the learning of multiple levels of composition.
- It is considered a more general principle that surpasses conventional/classic machine learning approaches.

Network Structure

- Feedforward networks consist of **layers** of interconnected **neurons**.
- Layers can be categorized into input, hidden, and output layers.
  - **Graph Representation:** Neural networks use an acyclic graph, like $f(\mathbf{x}) = f^{(3)}(f^{(2)}(f^{(1)}(\mathbf{x})))$, depicting the composition of functions.
  - **Depth and Output:** The chain's length determines depth in *deep learning*. The last layer, the output layer, matches and produces target outputs during training.
  - **Hidden Layers:** Situated between input and output, hidden layers are crucial for learning and approximating f^*. The algorithm decides their role for optimal approximation.
  - **Neuroscience Inspiration:** Neural networks borrow from neuroscience, using vector-valued hidden layers with width determined by dimensionality. Layers parallelly compute, mirroring the brain's structure.

Activation Functions

- **Non-Linearity Introduction:** Activation functions bring non-linearity, enabling the network to model complex relationships.
- **Common Choices:** ReLU, sigmoid, and tanh are widely used activation functions.
  - **Rectified Linear Unit (ReLU):** $f(x) = \max(0, x)$
  - **Sigmoid:** $f(x) = \frac{1}{1+e^{-x}}$
  - **Hyperbolic Tangent (tanh):** $f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
- **Custom Activation:** Depending on the problem, custom activation functions can be designed.
- **Activation Impact:** The choice influences the network's learning capacity and convergence speed.

Cost Function

- **Task Dependence:** The choice of cost function is task-specific.
- **Classification:** Cross-entropy is commonly used for classification tasks.
- **Regression:** Mean Squared Error (MSE) or Mean Absolute Error (MAE) are preferred for regression.
- **Customization:** Depending on the problem, custom cost functions can be designed.
- **Loss Landscape:** The cost function shapes the optimization landscape during training.
- **Trade-offs:** Selection involves trade-offs between robustness, sensitivity, and computational efficiency.

Gradient-Based Learning

- **Weight Tuning:** Feedforward network training involves adjusting weights to minimize the chosen cost function.
- **Non-Convex Challenges:** The cost function's non-convex nature poses optimization difficulties during training.
- **Local Minima:** Networks may converge to suboptimal local minima due to non-convexity.
- **Gradient Descent:** Gradient-based methods like backpropagation are commonly used for weight updates.
  - **Momentum**: GD updates parameters based only on the current gradient. Momentum considers both the current gradient and a historical moving average of gradients, improving convergence and reducing oscillations.
  - **Exploration Strategies:** Techniques like stochastic GD and adaptive learning rates address optimization challenges.

Forward Propagation and Backprop

- **Forward Propagation:** Input passes through layers, computing weighted sums, applying activations, and passing information to subsequent layers.
- **Backprop:** Backward propagation of errors, the core training algorithm for neural networks, involves:
  - **Error Calculation:** Compare predictions to target values using the chosen cost function.
  - **Gradient Descent:** Propagate error backward, computing gradients of the cost function with respect to weights and biases.
  - **Weight Updates:** Update weights and biases using computed gradients and an optimization algorithm, typically gradient descent, to minimize error and improve model performance.

Regularization Techniques

- **Norm Penalties:** Regularization terms are added to the loss function, penalizing large weights. L1 regularization enforces sparsity, while L2 regularization discourages overly complex models.
- **Early Stopping:** Training is halted when the model's performance on a validation set stops improving, preventing overfitting to the training data.
- **Dropout:** Randomly deactivates a fraction of neurons during training, promoting robustness and preventing reliance on specific neurons.
- **Data Augmentation:** Introduces variations in training data by applying transformations like rotation, scaling, or flipping, enhancing model generalization.
- **Batch Normalization:** Normalizes input data within each mini-batch, reducing internal covariate shift and improving convergence.

Convolution Neural Network

Convolution Operation

- **Convolution Overview**
  - The convolution operation is the cornerstone of CNNs, involving the application of a filter (kernel) to the input data.
  - Mathematically, convolution is expressed as:

$$(x * w)(t) = \int_{-\infty}^{\infty} x(a) \cdot w(t-a)\, da$$

    where x is the input and w is the filter.

- **Feature Extraction**
  - Convolution enables the network to learn hierarchical features by capturing local dependencies within the input data.
  - This process is essential for identifying patterns and structures in the data.

Motivation for Convolution

- **Sparse Interactions**:
  - Sparsity reduces memory requirements and computation, enhancing efficiency.
  - Allows capturing meaningful features with smaller kernels.
- **Parameter Sharing**:
  - Sharing parameters across input positions dramatically improves efficiency.
  - Convolution is equivariant to translation, providing consistent representations.
- **Equivariant Representations**:
  - Convolution exhibits equivariance to translation, useful for creating timelines in time series data.
  - Not naturally equivariant to scale or rotation, requiring additional mechanisms.

Pooling

- Convolutional layers operate in three stages:
  - Parallel convolutions produce linear activations.
  - Nonlinear activation functions (e.g., ReLU) in the detector stage.
  - Pooling modifies the output using summary statistics.
- Pooling achieves **invariance to small translations**.
  - Adds an **infinitely strong prior** for translation invariance.
  - Reduces computational and memory requirements.
  - Essential for handling inputs of varying size.
- Different pooling functions exist (e.g., max pooling, average pooling).
- Dynamic pooling and theoretical guidance for various situations.

Convolution & Pooling as Strong Priors

- Priors can be weak or strong based on probability density concentration.
- **Infinitely strong priors** strictly restrict some parameters.
- Convolution as a fully connected net with a strong prior:
- Promotes local interactions and equivariance to translation.
- Pooling as an **infinitely strong prior** for translation invariance.
- Can cause **underfitting** if assumptions are inaccurate.
- Models designed for balancing translation invariance and information preservation.
- Compare convolutional models to benchmarks of statistical learning performance.

Stride and Padding

- **Stride**:
  - Controls the step size of the filter during convolution.
  - Larger strides result in smaller output spatial dimensions.
  - Influences overlap, computational efficiency, and information preservation.
- **Padding**:
  - Padding adds extra border pixels to the input before convolution.
  - Mitigates spatial dimension reduction, preserving input information.
  - Common padding types include zero-padding.
- Which one is more useful in time series analysis?