

# Lecture: Deep Learning Fundamentals

KTH AI Student

February 19, 2025

# What we will cover today



- ▶ Fundamentals of Linear Regression.
- ▶ Understanding and Implementing Gradient Descent.
- ▶ Introduction to Autograd and Backpropagation.
- ▶ Training a Multilayer Perceptron (MLP).
- ▶ Deep Dive into Convolutional Neural Networks (CNNs).

# What is Linear Regression?



- ▶ A statistical method to model the relationship between a dependent variable  $y$  and an independent variable  $x$ .
- ▶ The equation for a simple linear model is:

$$y = wx + b$$

- ▶ Our goal is to find the best  $w$  and  $b$  that minimize error.

- ▶ Linearity: The relationship between dependent and independent variables is linear.
- ▶ Independence: Observations are independent of each other.
- ▶ Homoscedasticity: Constant variance of residuals.
- ▶ Normality: Residuals should be normally distributed.

# Loss Function for Linear Regression



- ▶ We measure the model's performance using **Mean Squared Error (MSE)**:

$$\text{MSE} = \frac{1}{N} \sum (y_i - (wx_i + b))^2$$

- ▶ The lower the MSE, the better the model fits the data.

# Gradient Descent Algorithm



- ▶ An optimization technique used to minimize the loss function.
- ▶ The parameter updates are:

$$w := w - \alpha \frac{\partial \text{MSE}}{\partial w}$$

$$b := b - \alpha \frac{\partial \text{MSE}}{\partial b}$$

- ▶  $\alpha$  is the learning rate, which controls step size.

# Challenges in Gradient Descent



- ▶ Choosing the correct learning rate is critical.
- ▶ Too small  $\alpha$  results in slow convergence.
- ▶ Too large  $\alpha$  may cause divergence.
- ▶ Local minima vs. global minima.

- ▶ **Batch Gradient Descent:** Uses the entire dataset to compute gradients.
- ▶ **Stochastic Gradient Descent (SGD):** Uses one sample at a time.
- ▶ **Mini-batch Gradient Descent:** Uses a subset of the dataset.
- ▶ **Momentum:** Accelerates convergence by considering past gradients.
- ▶ **Adam:** Adaptive learning rate method combining momentum and RMSProp.



# Mathematical Derivation of Gradient Descent



- ▶ The gradient of MSE with respect to  $w$  is:

$$\frac{\partial \text{MSE}}{\partial w} = -\frac{2}{N} \sum x_i(y_i - (wx_i + b))$$

- ▶ The gradient of MSE with respect to  $b$  is:

$$\frac{\partial \text{MSE}}{\partial b} = -\frac{2}{N} \sum (y_i - (wx_i + b))$$

- ▶ Update rules:

$$w := w - \alpha \left( -\frac{2}{N} \sum x_i(y_i - (wx_i + b)) \right)$$

$$b := b - \alpha \left( -\frac{2}{N} \sum (y_i - (wx_i + b)) \right)$$

# What is Autograd?



- ▶ Autograd automatically computes gradients using the **chain rule**.
- ▶ It enables efficient backpropagation for neural networks.
- ▶ Saves effort compared to manual differentiation.

# Backpropagation: The Core of Neural Networks



- ▶ Backpropagation is used to adjust weights to minimize loss.
- ▶ Steps:
  1. Compute forward pass (prediction).
  2. Calculate loss function.
  3. Compute gradients of loss with respect to parameters.
  4. Update weights using gradient descent.

# Chain Rule in Backpropagation



- ▶ The derivative of a composite function:

$$\frac{df}{dx} = \frac{df}{du} \times \frac{du}{dx}$$

- ▶ Used to efficiently propagate gradients in deep networks.

# Example of Backpropagation



- ▶ Consider a simple neural network with one hidden layer.
- ▶ Forward pass:

$$z_1 = w_1x + b_1$$

$$a_1 = \sigma(z_1)$$

$$z_2 = w_2a_1 + b_2$$

$$\hat{y} = \sigma(z_2)$$

- ▶ Loss function:

$$L = \frac{1}{2}(\hat{y} - y)^2$$

# Backpropagation Steps



- Compute gradients:

$$\frac{\partial L}{\partial \hat{y}} = \hat{y} - y$$

$$\frac{\partial \hat{y}}{\partial z_2} = \sigma'(z_2)$$

$$\frac{\partial z_2}{\partial w_2} = a_1$$

- Chain rule:

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_2} \cdot \frac{\partial z_2}{\partial w_2}$$

- Update weights:

$$w_2 := w_2 - \alpha \frac{\partial L}{\partial w_2}$$

# What is an MLP?



- ▶ A **Multilayer Perceptron (MLP)** consists of multiple layers of neurons.
- ▶ It includes an **input layer, hidden layers, and an output layer**.
- ▶ Uses **activation functions** to introduce non-linearity.

- ▶ **Sigmoid:** Used in binary classification problems.

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

- ▶ **ReLU (Rectified Linear Unit):**

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

- ▶ **Tanh:** Scales input to range  $[-1, 1]$ .

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



- ▶ Input layer: Receives input features.
- ▶ Hidden layers: Perform computations and transformations.
- ▶ Output layer: Produces final predictions.
- ▶ Example: MLP for digit classification.

- ▶ Initialize weights and biases.
- ▶ Forward pass: Compute predictions.
- ▶ Compute loss: Measure prediction error.
- ▶ Backward pass: Compute gradients.
- ▶ Update weights: Apply gradient descent.
- ▶ Repeat until convergence.

- ▶ **L2 Regularization:** Adds penalty for large weights.

$$L = \text{MSE} + \lambda \sum w^2$$

- ▶ **Dropout:** Randomly drops neurons during training.
- ▶ **Early Stopping:** Stops training when validation loss increases.

- ▶ 1943: McCulloch and Pitts propose the first mathematical model of a neuron.
- ▶ 1958: Rosenblatt introduces the Perceptron, an early neural network.



- ▶ 1970s: AI research faces criticism and reduced funding.
- ▶ Limitations of Perceptrons highlighted by Minsky and Papert.



- ▶ 1980s: Backpropagation algorithm popularized by Rumelhart, Hinton, and Williams.
- ▶ 1990s: AI research faces another decline due to unmet expectations.



- ▶ 1997: IBM's Deep Blue defeats world chess champion Garry Kasparov.
- ▶ 2006: Hinton and colleagues introduce deep belief networks, sparking renewed interest in deep learning.

- ▶ 2012: AlexNet wins ImageNet competition, demonstrating the power of deep convolutional networks.
- ▶ 2014: GANs introduced by Goodfellow et al., enabling generative models.
- ▶ 2017: Transformers introduced by Vaswani et al., revolutionizing NLP.



# What is a CNN?



- ▶ CNNs are specialized for image recognition and processing.
- ▶ Use **convolutional layers** to extract important features.
- ▶ Pooling layers reduce dimensionality while retaining information.

- ▶ Apply filters (kernels) to detect features like edges.
- ▶ Example: Using a **3x3 kernel** for edge detection.
- ▶ Convolution operation:

$$(I * K)(i, j) = \sum_m \sum_n I(i + m, j + n) K(m, n)$$

- ▶ Reduce the size of feature maps while preserving important features.
- ▶ **Max pooling** selects the highest value in a region.
- ▶ **Average pooling** computes the average value in a region.

- ▶ After convolution and pooling, a fully connected layer is used for classification.
- ▶ Example: Final layer in **image classification** predicts categories.

- ▶ Input layer: Receives image data.
- ▶ Convolutional layers: Extract features.
- ▶ Pooling layers: Reduce dimensionality.
- ▶ Fully connected layers: Perform classification.
- ▶ Example: LeNet-5 for digit recognition.

- ▶ Initialize filters and weights.
- ▶ Forward pass: Compute feature maps and predictions.
- ▶ Compute loss: Measure prediction error.
- ▶ Backward pass: Compute gradients.
- ▶ Update weights: Apply gradient descent.
- ▶ Repeat until convergence.

- ▶ Technique to increase the diversity of training data.
- ▶ Examples: Rotation, scaling, flipping, and cropping.
- ▶ Helps prevent overfitting and improves generalization.

- ▶ Use pre-trained models on large datasets.
- ▶ Fine-tune the model on a specific task.
- ▶ Example: Using VGG16 for image classification.



# Recurrent Neural Networks (RNNs)



- ▶ Designed for sequential data.
- ▶ Maintain hidden states to capture temporal dependencies.
- ▶ Applications: Time series prediction, language modeling.

# Long Short-Term Memory (LSTM)



- ▶ A type of RNN designed to handle long-term dependencies.
- ▶ Uses gates to control the flow of information.
- ▶ Applications: Speech recognition, text generation.

# Generative Adversarial Networks (GANs)



- ▶ Consist of a generator and a discriminator.
- ▶ Generator creates fake data, discriminator distinguishes real from fake.
- ▶ Applications: Image generation, data augmentation.

- ▶ Use self-attention mechanisms to process sequences in parallel.
- ▶ Revolutionized NLP tasks.
- ▶ Applications: Machine translation, text summarization.

# Large Language Models (LLMs)



- ▶ Built on transformer architecture.
- ▶ Trained on vast amounts of text data.
- ▶ Applications: Chatbots, content generation, question answering.

- ▶ **Linear Regression** is a fundamental prediction method.
- ▶ **Gradient Descent** optimizes model parameters.
- ▶ **Autograd** automates differentiation for neural networks.
- ▶ **MLPs** introduce complexity beyond linear models.
- ▶ **CNNs** excel in image-related tasks.
- ▶ **RNNs, LSTMs, GANs, Transformers, and LLMs** represent current advancements in deep learning.