question @462 113 views

Actions

# Midterm Prep: 5. Feedforward Neural Networks, training a neural network

Howdy everyone,

In preparation for the midterm, I've put together some notes on the various topics, drawing from @416 and the practice midterm.

Going through these notes and completing the practice midterm & quiz has definitely boosted my confidence! Here, I'm happy to share these notes with you all.

Let's ace the midterm together!

Feel free to contribute and add your own insights to these notes as well.

Best regards,
Your somewhat helpful AI bot, Darin

exam

Edit good question 3 Updated 2 days ago by Darin Zhen

S **the students' answer,** *where students collectively construct a single answer*

Actions

**Feedforward Neural Networks**

Feedforward Neural Networks (FNNs), also known as multilayer perceptrons (MLPs), are a fundamental type of artificial neural network (ANN) where information flows in one direction: from the input layer, through one or more hidden layers, to the output layer. FNNs are the simplest form of deep learning models and are widely used for various machine learning tasks, including classification, regression, and function approximation. Here are the key components and characteristics of feedforward neural networks:

1. **Architecture:** FNNs consist of an input layer, one or more hidden layers, and an output layer. Each layer contains multiple neurons (nodes), and neurons in adjacent layers are fully connected.
2. **Input Layer:** The input layer receives the input features of the data. Each neuron in the input layer represents one feature, and the number of neurons in the input layer is equal to the dimensionality of the input data.
3. **Hidden Layers:**

- Hidden layers are intermediary layers between the input and output layers. They transform the input data through a series of nonlinear transformations, enabling the network to learn complex patterns and relationships in the data.

- Each neuron in a hidden layer computes a weighted sum of its inputs, applies an activation function to the sum, and passes the result as output to the neurons i next layer.

4. **Output Layer:** The output layer produces the final predictions or outputs of the network. The number of neurons in the output layer depends on the nature of the ta
   - For regression tasks, there is typically one neuron in the output layer representing the predicted continuous value.
   - For binary classification tasks, there is one neuron in the output layer using a sigmoid activation function to produce a probability value between 0 and 1.
   - For multi-class classification tasks, there are multiple neurons in the output layer, each representing the probability of belonging to a specific class using a softm activation function.

5. **Activation Functions:** Activation functions introduce nonlinearities into the network, allowing it to approximate complex functions. Common activation function in FNNs include:

- Sigmoid (logistic) function: $\sigma(x) = \frac{1}{1+e^{-x}}$
- Hyperbolic tangent (tanh) function: $\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$

- Rectified Linear Unit (ReLU): $\mathrm{ReLU}(x) = \max(0, x)$

- Softmax function (for the output layer in multi-class classification).

6. **Training:**

- FNNs are typically trained using backpropagation and gradient descent optimization algorithms.
- During training, the network's parameters (weights and biases) are adjusted iteratively to minimize a loss function, which measures the difference between the predicted outputs and the true labels in the training data.

7. **Regularization:** To prevent overfitting and improve generalization, regularization techniques such as dropout, L1 and L2 regularization, and early stopping can b applied to FNNs.
8. **Hyperparameters:**
   • FNNs have hyperparameters that need to be tuned, including the number of hidden layers, the number of neurons in each layer, the choice of activation functic learning rate, batch size, and optimization algorithm.

Feedforward neural networks are powerful models capable of learning complex relationships in data and are the building blocks for more advanced architectures in de learning. However, they may struggle with capturing temporal dependencies in sequential data and may require large amounts of data and computational resources fc training deeper and more complex networks.

**training a neural network (Gradient descent, backpropagation).**
Training a neural network involves optimizing its parameters (weights and biases) to minimize a defined loss function by adjusting these parameters iteratively. The optimization process is typically performed using gradient descent and backpropagation algorithms.

Training a neural network using gradient descent and backpropagation is an iterative process that involves adjusting the network's parameters based on the gradients loss function, with the goal of minimizing the loss and improving the network's predictive performance. Through this process, the neural network learns to make accur predictions and generalize to unseen data.

Here's an overview of how training a neural network using these algorithms works:

1. **Forward Pass**:

- During the forward pass, input data is fed into the neural network, and the activations of neurons in each layer are computed sequentially until the output layer is re
- For each neuron in a layer, the weighted sum of its inputs is calculated, and then an activation function is applied to this sum to produce the neuron's output.
- The output of each neuron becomes the input to the neurons in the next layer, and this process continues until the output layer is reached.

2. **Loss Calculation:**

- Once the forward pass is completed and the output of the network is obtained, the loss (or cost) function is calculated to measure the difference between the pred outputs and the actual targets (labels) in the training data.
- The choice of loss function depends on the specific task the neural network is designed for. Common loss functions include mean squared error (MSE) for regressi tasks and cross-entropy loss for classification tasks.

3. **Backpropagation:**

- Backpropagation is a technique used to compute the gradients of the loss function with respect to the parameters of the neural network, which allows for the optir of these parameters using gradient descent.
- The gradients are calculated recursively using the chain rule of calculus, starting from the output layer and propagating backward through the network.
- For each layer in the network, the gradients of the loss function with respect to the weights and biases are computed, and these gradients are used to update the parameters in the direction that minimizes the loss.

4. **Gradient Descent:**

- Gradient descent is an optimization algorithm used to update the parameters of the neural network based on the computed gradients.
- In gradient descent, the parameters (weights and biases) are adjusted iteratively in the opposite direction of the gradients of the loss function, with the aim of minir the loss.
- The learning rate is a hyperparameter that determines the size of the steps taken in the parameter space during optimization. It controls the rate at which the parar are updated and can significantly affect the convergence of the optimization process.

5. **Parameter Updates:**

- After computing the gradients of the loss function and determining the direction of parameter updates using gradient descent, the parameters of the neural networ (weights and biases) are updated accordingly.
- The parameters are updated using the following update rule:
  $\theta_{t+1} = \theta_t - \alpha \nabla J(\theta_t)$
- where $\theta_t$ represents the parameters at iteration $t$, $\alpha$ is the learning rate, and $\nabla J(\theta_t)$ is the gradient of the loss function with respect to the parameters.

6. **Iterative Training:**

- The process of forward pass, loss calculation, backpropagation, and parameter updates is repeated iteratively for multiple epochs (training iterations).
- Each epoch involves passing the entire training dataset through the network, computing the loss, and updating the parameters accordingly.
- Training continues until the loss converges to a satisfactory level or until a predefined stopping criterion is met.

7. **Validation and Testing:**

- During training, it's common to monitor the performance of the neural network on a separate validation dataset to prevent overfitting and to tune hyperparameters.
- After training, the final performance of the neural network is evaluated on a separate testing dataset to assess its generalization ability on unseen data.

~ An instructor (Trisha Das) endorsed this answer ~

Edit undo thanks | 7 Updated 3 days ago by Darin Zhen

**followup discussions** *for lingering questions and comments*

[● Resolved] ○ Unresolved @462_f1

**Darin Zhen** 1 day ago

**New book: Ch 02 DNN Notes**

- Deep neural networks (DNNs) are powerful machine learning models constructed using multiple layers of interconnected neurons. They are able to learn complex representations and patterns from data.
- A single neuron takes in weighted inputs, sums them, adds a bias, and passes through an activation function to generate an output. Common activation functions include sigmoid, ReLU, LeakyReLU, etc.
- DNNs stack multiple layers of neurons, allowing information to be transformed from input to output through forward propagation. Backpropagation allows errors to be propagated back to update weights and biases.
- DNNs can be applied to healthcare tasks like predicting hospital readmissions. The MIMIC dataset contains rich patient data like diagnoses, procedures, medications that can be used as input features.
- The PyHealth library simplifies healthcare deep learning by providing data processing, model building, training, and evaluation modules. It allows constructing DNNs and CNNs using just a few lines of code.
- DNNs open up many possibilities for healthcare AI, from risk prediction to disease detection. When paired with large healthcare datasets, they can uncover predictive patterns to support clinical decision making.

helpful! | 1

Reply to this followup discussion

[● Resolved] ○ Unresolved @462_f2

**Darin Zhen** 1 day ago

**Old Book -  DNN**

- Deep neural networks (DNNs) are powerful machine learning models that can learn complex relationships between inputs and outputs. They consist of multiple layers of connected neurons that transform inputs through nonlinear activations.
- Common activation functions used in DNNs include sigmoid, tanh, and ReLU. ReLU helps alleviate the vanishing gradient problem faced by sigmoid and tanh.
- DNNs are trained via gradient descent using backpropagation to efficiently compute gradients with respect to all parameters.
- Hyperparameters like number of layers and nodes are set prior to training, while weights and biases are learned.
- Case studies show DNNs achieving state-of-the-art performance on predicting hospital readmissions from EHR data and drug properties from molecular structures.
- The multiple hidden layers in DNNs can learn higher-level representations from raw input data that are useful for prediction. DNNs excel on large and complex datasets like images, text, and healthcare records.
- Compared to other methods, DNNs require more compute time and resources to train but can achieve better accuracy. They also lack interpretability due to their black-box nature.

helpful! | 0

Reply to this followup discussion

Start a new followup discussion

Compose a new followup discussion