optilearn.nn.ReduceLearningRate

08 November 2024 12:47

ReduceLearningRate

A custom callback class that reduces the learning rate based on the lack of improvement in a specified metric. This class allows users to adjust the learning rate during training when improvements in performance metrics have plateaued, helping to achieve more efficient training and avoiding overfitting.

Attributes

- monitor : str
  - Metric to monitor for improvement. Available choices are: 'accuracy', 'val_accuracy', 'loss', and 'val_loss'.
- factor : float
  - Factor by which the learning rate will be reduced. The new learning rate is computed as: new_lr = lr * factor.
- patience : int
  - Number of epochs with no improvement before reducing the learning rate.
- verbose : int
  - Verbosity mode (0 or 1). Set to 1 to display a message when the learning rate is reduced.
- min_delta : float
  - Minimum change in the monitored metric to qualify as an improvement.
- cooldown : int
  - Number of epochs to wait after reducing the learning rate before resuming normal operation.
- min_lr : float
  - Minimum learning rate limit after which it will not be reduced further.

Example Usage

```
# Create an instance with specific parameters
reduce_lr_callback = ReduceLearningRate(monitor='val_loss', factor=0.5, patience=5,
```

__init__(self, monitor='val_accuracy', factor=0.1, patience=10, verbose=0, min_delta=1e-4, cooldown=0, min_lr=0)
The constructor for initializing ReduceLearningRate with specified parameters.

Parameters:

- monitor : str, optional
  - Metric to monitor. Default is 'val_accuracy'.
- factor : float, optional
  - Multiplicative factor for learning rate reduction. Default is 0.1.
- patience : int, optional
  - Number of epochs with no improvement after which the learning rate will be reduced. Default is 10.
- verbose : int, optional
  - Verbosity level, set to 1 to print updates. Default is 0.
- min_delta : float, optional
  - Minimum change to qualify as an improvement. Default is 1e-4.
- cooldown : int, optional
  - Number of epochs to wait after reducing learning rate. Default is 0.
- min_lr : float, optional
  - Minimum learning rate allowed. Default is 0.

check(self, metrics_values, number_of_epoch, pre_value, time_step, current_learning_rate, cooldown_timestep)
Evaluates whether to reduce the learning rate based on the monitored metric.

- **Parameters:**
  - metrics_values : dict
    - Dictionary containing the metric values with monitored metric as a key.
  - number_of_epoch : int
    - Current epoch number in training.
  - pre_value : float
    - Value of the monitored metric from the previous epoch.
  - time_step : int
    - Counter for epochs with no metric improvement.
  - current_learning_rate : float
    - Current learning rate.
  - cooldown_timestep : int
    - Counter for cooldown period after last reduction.
- **Returns:**
  - tuple
    - A tuple containing:
      - int : Reduction flag (1 if learning rate was reduced, otherwise 0).
      - float : Updated learning rate.
      - float : Last monitored metric value that did not improve.
      - int : Updated time step for epochs without improvement.
      - int : Updated cooldown time step.

Example Workflow

Suppose the model is trained for 100 epochs, with ReduceLearningRate configured as:

```
ReduceLearningRate(monitor='val_loss', factor=0.5, patience=3, cooldown=2, min_lr=1e-5)
```

1. **Initial Epochs (0-3)**:
   - Monitors val_loss. If improvement, resets the time step. Otherwise, increments time step.
2. **Epoch 4**:
   - If no improvement since the 1st epoch, time step reaches patience of 3. Reduces learning rate by factor (0.5) and initiates cooldown (2 epochs).

3. **Cooldown Period**:
    - Checks val_loss but skips any learning rate change until cooldown is complete.
4. **Post-Cooldown**:
    - If no improvement within cooldown, repeats reduction and cooldown.

This setup optimizes learning by dynamically reducing the learning rate, enhancing model performance, and avoiding overfitting through controlled metric tracking.