STEP32. 고차 미분 (구현 편)

고차 미분 구현
- 역전파 시 수행되는 계산에 대해서 계산 그래프를 만들면 됨
- 역전파 시에도 Variable 인스턴스를 사용하면 해결됨

패키지 구조 변경

- 지금까지는 Variable 클래스를 dezero/core_simple.py에 구현함
- 고차 미분을 할수 있는 새로운 Variable 클래스를 dezero/core.py 에 구현
- dezero/core_simple.py에 구현했던 사칙연산 등의 함수와 연산자 오버로드들 또한 dezero/core.py 에서 새로 구현함

새로운 DeZero로!

1. 새로운 DeZero의 가장 중요한 변화

```
class Variable:

  ```

  # retain_grad=False : 중간 변수의 미분값을모두 None으로 재설정
  def backward(self, retain_grad=False):
      # y.grad = np.array(1.0) 생략을 위한 if문
      if self.grad is None:
          """ Variable 인스턴스 참조 """
          self.grad = Variable(np.ones_like(self.data))

  ```
```

- Variable 클래스의 인스턴스 변수인 grad 임
- 기존 grad는 ndarray 인스턴스를 참조 했는데, 새로운 클래스에서는 Variable 인스턴스를 참조
- Variable 클래스의 소스 변경
  - 미분값을 자동으로 저장하는 코드에서 self.grad 가 Variable 인스턴스를 담게 됨

함수 클래스의 역전파

backward 메서드 수정

- Function 클래스는 수정할 것이 없음
- Add, Mul, Neg, Sub, Div, Pow 클래스의 backward 메서드 수정

함수 클래스의 역전파 구현

- Add 클래스의 역전파가 하는 일은 출력 쪽에서 전해지는 미분값을 입력 쪽으로 전달 (역전파 때는 아무것도 계산하지 않기 때문에 수정할 것이 없음)
- Mul 클래스의 역전파 수정
  - 수정 전에는 Variable 인스턴스 안에 있는 데이터를 꺼내야 했음
  - 수정 후에는 Mul 클래스에서 Variable 인스턴스를 그대로 사용함
  - 역전파를 계산하는 gy * x1 코드에서 gy와 x1이 Variable 인스턴스 임
  - gy*x1 이 실행되는 뒤편에서 Mul 클래스의 순전파가 호출되면서 그 때 `Function.__call__` 이 호출되고, 그 안에서 계산 그래프가 만들어짐

```
class Mul(Function): # 곱하기
    def forward(self, x0, x1):
        y = x0 * x1
        return y

    def backward(self, gy):
        """ Variable 인스턴스인 x0, x1 가져옴 """
        x0, x1 = self.inputs
        return gy * x1, gy * x0
```

역전파를 더 효율적으로 (모드 추가)

역전파의 활성/비활성 모드 도입

```
class Variable:

  ```
```
```

```
    def backward(self, retain_grad=False, create_graph=False):

      ```
      while funcs:
        f = funcs.pop()# 함수를 가져온다.
        gys = [output().grad for output in f.outputs]  # output is weakref
        gxs = f.backward(*gys) # 함수 f의 역전파 호출 ( 리스트 언팩 )

        with using_config('enable_backprop', create_graph):
            """ 메인 backward """
            gxs = f.backward(*gys)
            # gxs가 튜플이 아니라면 튜플로 변환
            if not isinstance(gxs, tuple):
                gxs = (gxs,)

            # 역전파로 전파되는 미분값을 Variable인스턴스 변수 grad에 저장
            for x, gx in zip(f.inputs, gxs):
                if x.grad is None:
                    x.grad = gx
                else:
                    x.grad = x.grad + gx

                if x.creator is not None:
                    add_func(x.creator)
```

- 역전파가 필요없는 경우는 역전파 비활성 모드로 전환하여 역전파 처리 생략
- 역전파를 1회만 한다면 역전파 계산도 역전파 비활성 모드로 실행하도록 함
- Variable 클래스의 backward 메서드에 다음 코드 추가
- create_graph를 추가
  - 기본값을 False로 설정
  - 2차 이상의 미분이 필요하다면 True로 설정
  - 실무에서 역전파가 단 1회만 수행되는 경우가 많음
- 역전파 처리
  - with using_config(...) 에서 수행

init.py 변경

지금까지의 수정 사항 반영

- 수정사항을 dezero/core.py 에 넣음
- 앞으로는 dezero/core_simple.py 대신 dezero/core.py를 사용함

초기화 수행

- is_simple_core 플래그 설정
- is_simple_core 를 False 로 설정하여 dezero/core.py에서 임포트하도록 변경
- 고차 미분에 대응하는 core 파일을 임포트할 수 있음
```