SAVE AND LOAD MODEL

一、内容

本部分将实现模型的两种保存和加载。

二、代码

一、保存参数

修改Layer_Dense

```
class Layer_Dense:
	def get_paramter(self):
		return self.weight, self.bias

	def set_paramter(self, weight, bias):
		self.weight = weight
		self.bias = bias
```

> 增加了get_paramter和load_paramter方法，用于dense层返回和加载参数.

修改Model

```
class Model():
	def get_paramter(self):
		paramter = []
		for layer in self.trainable_layer:
			paramter.append(layer.get_paramter())
		return paramter

	def set_paramter(self, paramter):
		for paramter_set, layer in zip(paramter, self.trainable_layer):
			layer.set_paramter(*paramter_set)
```

```
    def save_paramter(self, path):
        with open(path, 'wb') as f:
            pickle.dump(self.get_paramter(), f)

    def load_paramter(self, path):
        with open(path, 'rb') as f:
            self.set_paramter(pickle.load(f))
```

增加了get_paramter、set_paramter、save_paramter和load_paramter方法，用于整个模型层返回和加载参数。

实例1

```
X, y, X_test, y_test = data_preprocess()
print(X.shape, X_test.shape)


model = Model()
model.add(Layer_Dense(X.shape[1], 64, weight_L2=5e-4,bias_L2=5e-4))#,weight_L2=5e-
4,bias_L2=5e-4
model.add(Activation_ReLu())
model.add(Layer_Dense(64, 64))
model.add(Activation_ReLu())
model.add(Layer_Dense(64, 10))
model.add(Activation_Softmax())
model.set(loss=Loss_CategoricalCrossentropy(),
          optimizer=Optimizer_Adam(decay=5e-7),
          accuracy=Accuracy_Classification())


model.finalize()

model.train(X, y, batch_size=100, validation_data=(X_test, y_test), epochs=5,
print_every=10)
model.evaluate(X_test, y_test, batch_size=10)
# 反回各类别的概率
confidence = model.predict(X_test[95:105])
prediction = model.output_layer.prediction(confidence)

print('预测分类：', prediction)
print('ground truth：', y_test[95:105])


######################################################################
# 重新加载新模型
# 获得参数
paramter = model.get_paramter()
```

```
model = Model()
model.add(Layer_Dense(X.shape[1], 64, weight_L2=5e-4,bias_L2=5e-4))#,weight_L2=5e-4,bias_L2=5e-4
model.add(Activation_ReLu())
model.add(Layer_Dense(64, 64))
model.add(Activation_ReLu())
model.add(Layer_Dense(64, 10))
model.add(Activation_Softmax())
model.set(loss=Loss_CategoricalCrossentropy(),
                optimizer=Optimizer_Adam(decay=5e-7),
                accuracy=Accuracy_Classification())

model.finalize()

# 加载参数
model.set_paramter(paramter)

model.evaluate(X_test, y_test, batch_size=10)
#############################################################################
```

实例2

```
model.save_paramter('Mode_paramter.para')
model.load_paramter('Mode_paramter.para')
```

> 通过将参数保存到Mode_paramter.para文件，并通过文件加载参数。（文件后缀可以任意）

```
training 5, acc: 0.879, loss: 0.407 (data_loss: 0.329, reg_loss: 0.078), lr: 0.000998502745133672
validation, acc: 0.860, loss: 0.379
validation, acc: 0.860, loss: 0.379
预测分类： [[1]
 [3]
 [5]
 [3]
 [0]
 [0]
 [1]
 [4]
 [5]
 [8]]
ground truth： [1 3 5 3 6 1 1 4 5 8]
validation, acc: 0.860, loss: 0.379
```

> 第一个模型和加载后的模型在测试集上表现一样。

二、保存整个模型

```
class Model():
    def save_Model(self,path):
        model = copy.deepcopy(self)

        # 删除无关参数，减小模型大小
        # 减少模型文件的大小并提高保存和加载模型的效率
        model.loss.clean_cumulate()
        model.accuracy.clean_cumulate()
        model.input_layer.__dict__.pop('output', None)
        model.loss.__dict__.pop('dinput',None)

        for layer in model.layer:
            for property in ['input','output','dinput','dweight','dbias']:
                layer.__dict__.pop(property, None)

        with open(path, 'wb') as f:
            pickle.dump(model,f)

    # 不需要先实例化一个模型对象就能调用load方法
    @staticmethod
    def load_Model(path):
        with open(path, 'rb') as f:
            model = pickle.load(f)
        return model
```

增加了save_Model和load_Model方法。使用 @staticmethod 装饰器。这个装饰器可以与类方法一起使用，在未初始化的对象上运行它们，其中 self 不存在（注意它在函数定义中缺失）。在的例子中，将使用它来立即创建一个模型对象，而不需要先实例化一个模型对象。在这个方法中，将使用传入的路径以二进制读取模式打开一个文件，并使用 pickle 反序列化保存的模型。

```
training 5, acc: 0.881, loss: 0.402 (data_loss: 0.324, reg_loss: 0.078), lr: 0.000998502745133672
validation, acc: 0.866, loss: 0.374
validation, acc: 0.866, loss: 0.374
预测分类： [[7]
 [8]
 [7]
 [4]
 [4]
 [8]
 [8]
 [5]
 [7]
 [8]]
ground truth： [7 8 5 4 4 8 8 5 5 8]
validation, acc: 0.866, loss: 0.374
```

第一个模型和加载后的模型在测试集上表现一样。