人工智能基础 Lab2 实验报告

黄瑞轩 PB20111686

1 贝叶斯网络手写数字识别

1.1 代码原理

需要完成的部分是 `fit` 和 `predict` 两个函数，分别对应训练过程和预测过程，这里将代码原理以注释的形式给出。

训练过程的代码如下：

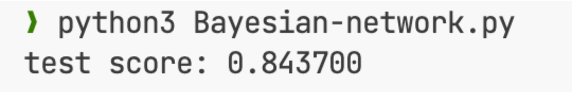
```python
# fit
for label in range(self.n_labels):
    # 计算每个类别的先验概率（即在样本中出现的频率）
    self.labels_prior[label] = np.sum(labels == label) / n_samples
    # 每个像素值在给定类别下的条件概率（即在给定类别的情况下，该像素值出现的频率）
    for pixel in range(self.n_pixels):
        for value in range(self.n_values):
            self.pixels_cond_label[pixel, value, label] = np.sum((pixels[:, pixel] == value) & (labels == label)) / np.sum(labels == label)
# 每个像素值的先验概率（即在样本中出现的频率）
for pixel in range(self.n_pixels):
    for value in range(self.n_values):
        self.pixels_prior[pixel, value] = np.sum(pixels[:, pixel] == value) / n_samples
```

预测过程的代码如下：

```python
# predict
n_samples = len(pixels)
labels = np.zeros(n_samples)
for i in range(n_samples):
    # 遍历所有可能的类别
    label_probs = np.zeros(self.n_labels)
    for label in range(self.n_labels):
        # 计算当前类别的先验概率
        label_probs[label] = self.labels_prior[label]
        # 计算当前类别下每个像素的条件概率
        for pixel in range(self.n_pixels):
            label_probs[label] *= self.pixels_cond_label[pixel, pixels[i, pixel], label]
        # 计算当前类别下每个像素的先验概率
        for pixel in range(self.n_pixels):
            label_probs[label] *= self.pixels_prior[pixel, pixels[i, pixel]]
    # 选择具有最大概率的类别
    labels[i] = np.argmax(label_probs)
return labels
```

1.2 运行结果

在测试集上的准确率为 84.37%



```
❱ python3 Bayesian-network.py
test score: 0.843700
```

2 利用 K-means 实现图片压缩

2.1 代码原理

需要完成的部分是 `assign_points`、`update_centers`、`fit` 和 `Compress`，分别对应簇分配、簇中心更新、整体训练过程和图像压缩过程。

簇分配的代码原理如下：

```
n_samples, n_dims = points.shape
labels = np.zeros(n_samples)
# 计算points离所有centers的距离，将points[i]分配给最近的centers[j]
for i in range(n_samples):
    distances = np.linalg.norm(points[i] - centers, axis=1)
    labels[i] = np.argmin(distances)
return labels
```

簇中心更新的代码原理如下：

```
# 以当前簇中所有样本的平均值作为新的簇中心
new_centers = np.zeros_like(centers)
for k in range(self.k):
    new_centers[k] = points[labels == k].mean(axis=0)
return new_centers
```

整体训练过程的代码原理如下：

```
n_samples, n_dims = points.shape
# 初始化一个簇中心数组
centers = self.initialize_centers(points)
for _ in range(self.max_iter):
    # 进行一次簇分配
    labels = self.assign_points(centers, points)
    # 更新簇中心
    new_centers = self.update_centers(centers, labels, points)
    # 检查是否收敛
    if np.all(centers == new_centers):
        break
    centers = new_centers
return centers
```

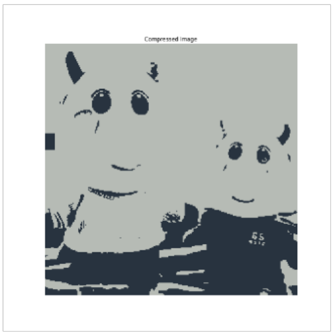
图像压缩过程的代码原理如下:

```
width, height, _ = img.shape
# 把图像转换成二维数组
img = img.reshape(width * height, -1)
# 使用k-means算法
centers = self.fit(img)
# 根据k-means算法得出的簇中心，进行一次分配
labels = self.assign_points(centers, img)
# 把所有像素的值修改为其簇中心的值
compressed_img = np.zeros_like(img)
for i in range(self.k):
    compressed_img[labels == i] = centers[i]
# 把图像转换为原来的维度
compressed_img = compressed_img.reshape(width, height, -1)
return compressed_img
```

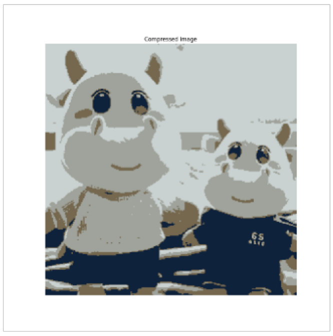
2.2 运行结果

original image

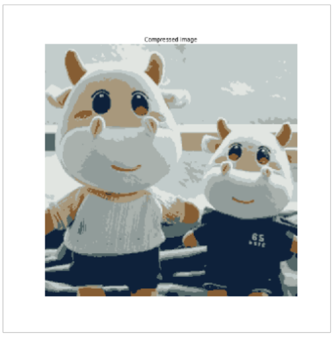
$k = 2$



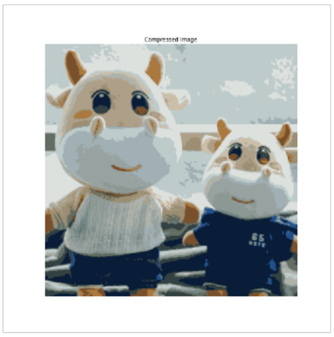
$k = 4$



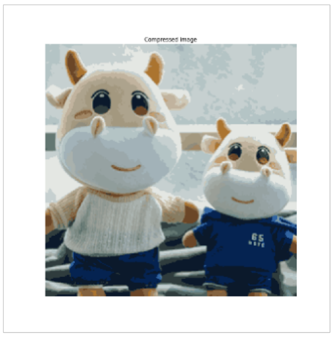
$k = 8$



$k = 16$



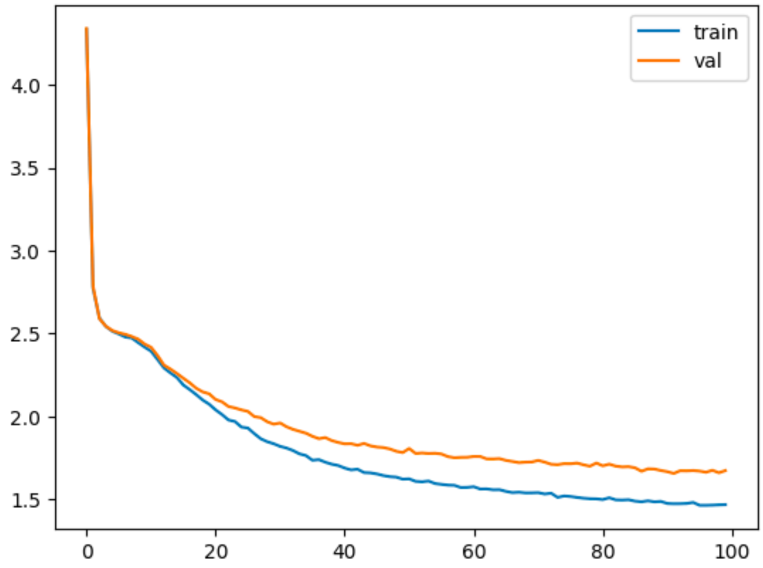
$k = 32$



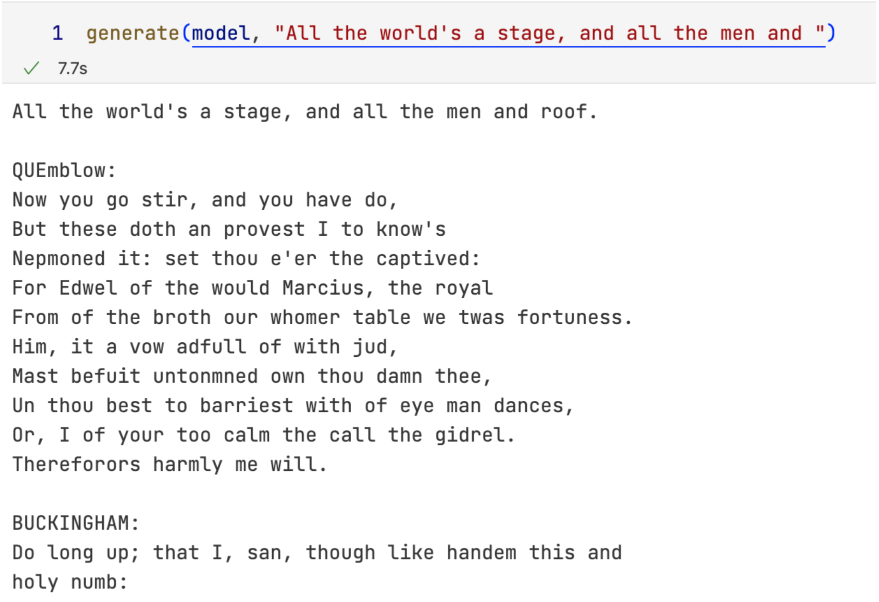
3　深度学习

3.1　误差变化

在训练过程中，训练误差和验证误差随迭代次数的变化如下：



3.2　补全测试



```
generate(model, "All the world's a stage, and all the men and ")
```

```
All the world's a stage, and all the men and roof.

QUEmblow:
Now you go stir, and you have do,
But these doth an provest I to know's
Nepmoned it: set thou e'er the captived:
For Edwel of the would Marcius, the royal
From of the broth our whomer table we twas fortuness.
Him, it a vow adfull of with jud,
Mast befuit untonmned own thou damn thee,
Un thou best to barriest with of eye man dances,
Or, I of your too calm the call the gidrel.
Therefor ors harmly me will.

BUCKINGHAM:
Do long up; that I, san, though like handem this and
holy numb:
```