# 模式识别与机器学习 -- 实验2

**姓名**：蔡亦扬
**学号**：23307130258

---

本实验包含以下部分：

- softmax （50%）
- svm （50%）

## softmax

1 手动实现 Softmax 函数 （15%）

```
def my_softmax(logits):
    # 数值稳定：减去每行最大值再求 exp，最后按行归一化
    max_per_row = logits.max(dim=-1, keepdim=True).values
    shifted_logits = logits - max_per_row
    exp_logits = torch.exp(shifted_logits)
    probs = exp_logits / exp_logits.sum(dim=-1, keepdim=True)
    return probs
```

说明：避免指数溢出，输出与 `torch.softmax` 一致，测试用例验证无 NaN/Inf。

2 创建自定义 Softmax 层 （15%）

```
class MySoftmax(nn.Module):
    """
    自定义 Softmax 层，使用手写 my_softmax
    """
    def __init__(self, dim=-1):
        super().__init__()
        self.dim = dim

    def forward(self, x):
        return my_softmax(x)
```

说明：封装手写 softmax 方便在模型末端输出概率，配合 `NLLLoss`。

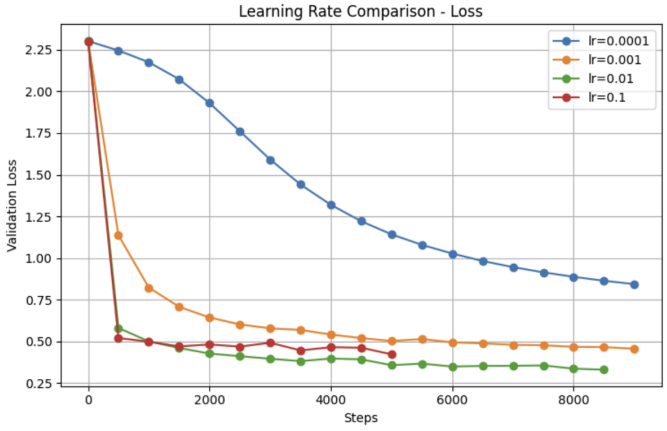
3 参数调优实验（无需给出代码）

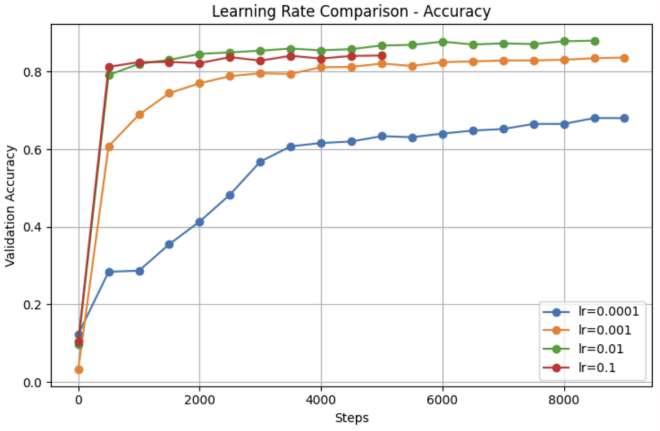
```
实验结果汇总:
       Experiment Parameter Best Val Acc Final Val Acc Final Val Loss
0   Learning Rate    0.0001       0.6805        0.6802         0.8438
1   Learning Rate     0.001       0.8365        0.8365         0.4566
2   Learning Rate      0.01       0.8800        0.8800         0.3307
3   Learning Rate       0.1       0.8417        0.8417         0.4228
4      Batch Size        16       0.8457        0.8457         0.4300
5      Batch Size        32       0.8621        0.8565         0.3904
6      Batch Size        64       0.8677        0.8534         0.3868
7      Batch Size       128       0.8735        0.8735         0.3481
8    Architecture       小网络         0.8787          0.8787           0.3389
9    Architecture       中网络         0.8728          0.8705           0.3533
10   Architecture       大网络         0.8657          0.8623           0.3785
11      Optimizer       SGD       0.8376        0.8339         0.4632
12      Optimizer      Adam       0.8708        0.8519         0.4182

结果已保存到 experiment_summary.csv
```

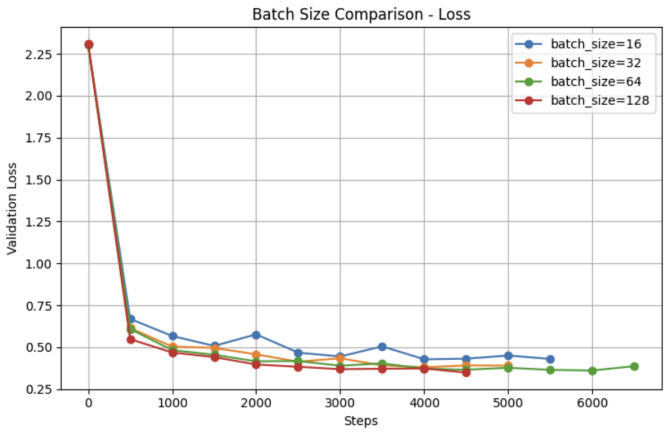
- 已实现 4 组实验函数：学习率、批次大小、网络结构、优化器对比（统一 `training` 流程）。
- 运行结果（来自 `softmax_and_tuning_dbc.ipynb`）：
  - 学习率：0.01 表现最佳，验证集准确率≈0.880，最终 val_loss≈0.3307。
  - 批次大小：128 最佳，验证集准确率≈0.8735，val_loss≈0.3481。
  - 网络结构：小网络(128,64) 最佳，验证集准确率≈0.8787，val_loss≈0.3389。
  - 优化器：Adam 略优于 SGD，验证集准确率≈0.8708，val_loss≈0.4182。

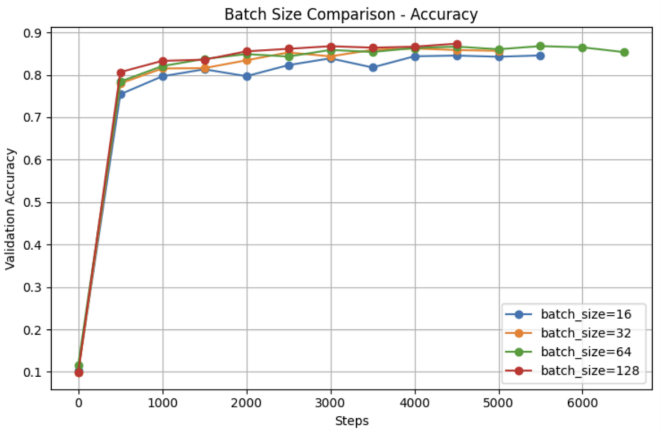
4 提交实验结果

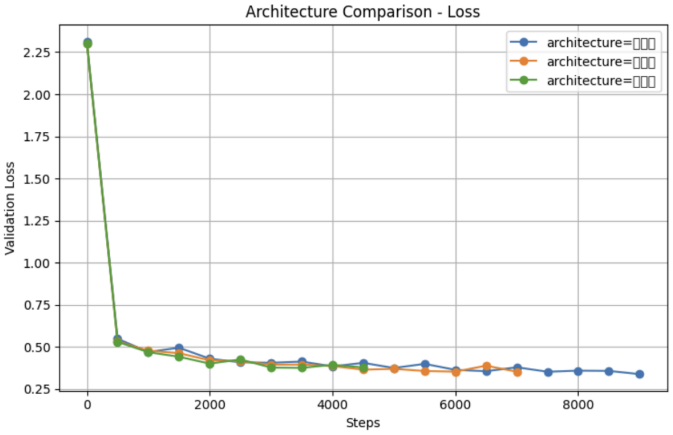
- 最佳模型训练（train_best_model）：lr=0.01，batch_size=128，结构(128,64)，优化器 Adam，早停于 epoch 19，最终验证准确率≈0.8753。

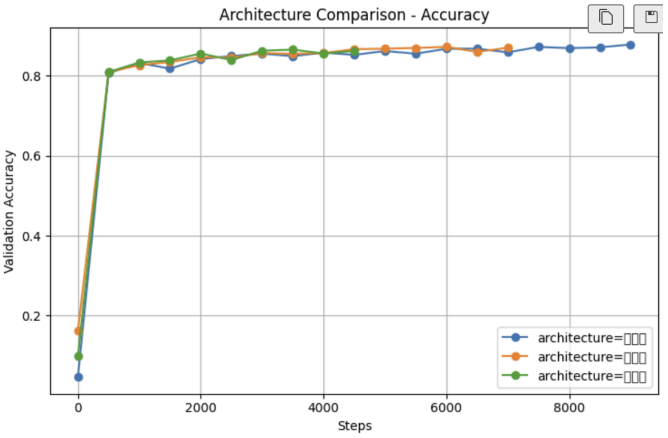

Learning Rate Comparison - Loss
Validation Loss
Steps
lr=0.0001
lr=0.001
lr=0.01
lr=0.1

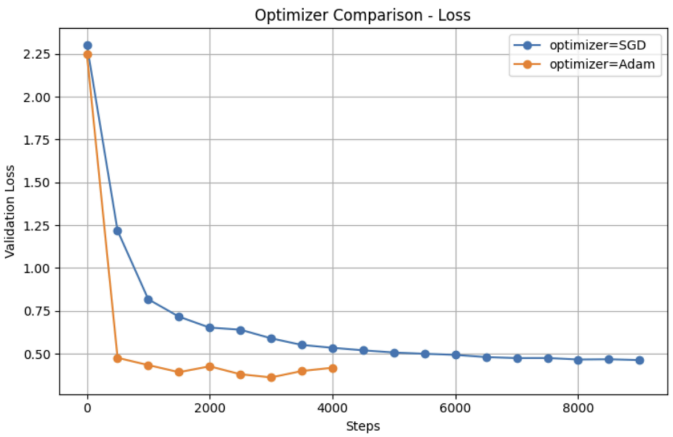

Learning Rate Comparison - Accuracy
Validation Accuracy
Steps
lr=0.0001
lr=0.001
lr=0.01
lr=0.1

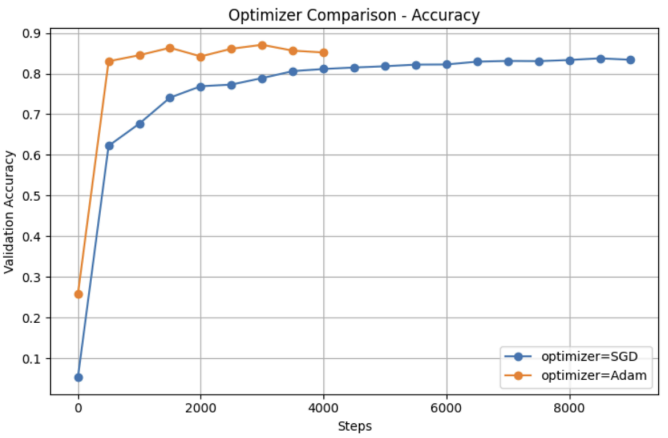

Batch Size Comparison - Loss
Validation Loss
Steps
batch_size=16
batch_size=32
batch_size=64
batch_size=128


Batch Size Comparison - Accuracy
Validation Accuracy
Steps
batch_size=16
batch_size=32
batch_size=64
batch_size=128


Architecture Comparison - Loss
Validation Loss
Steps
architecture=□□□
architecture=□□□
architecture=□□□


Architecture Comparison - Accuracy
Validation Accuracy
Steps
architecture=□□□
architecture=□□□
architecture=□□□


Optimizer Comparison - Loss
Validation Loss
Steps
optimizer=SGD
optimizer=Adam


Optimizer Comparison - Accuracy
Validation Accuracy
Steps
optimizer=SGD
optimizer=Adam

# svm

## 一、损失和梯度的计算

### 1, 循环实现中的梯度计算（10%）

补全 `fduml.linear_svm` 中的 `svm_loss_naive`：

```
for i in range(num_train):
    scores = X[i].dot(W)
    correct_class_score = scores[y[i]]
    for j in range(num_classes):
        if j == y[i]:
            continue
        margin = scores[j] - correct_class_score + 1
        if margin > 0:
            dW[:, j] += X[i]
            dW[:, y[i]] -= X[i]
dW /= num_train
dW += 2 * reg * W
```

说明：当 margin>0 时，对错误类累加特征，对正确类扣减特征；平均后加 L2 正则梯度。

### 2, 向量实现中的损失计算和梯度计算（15%）

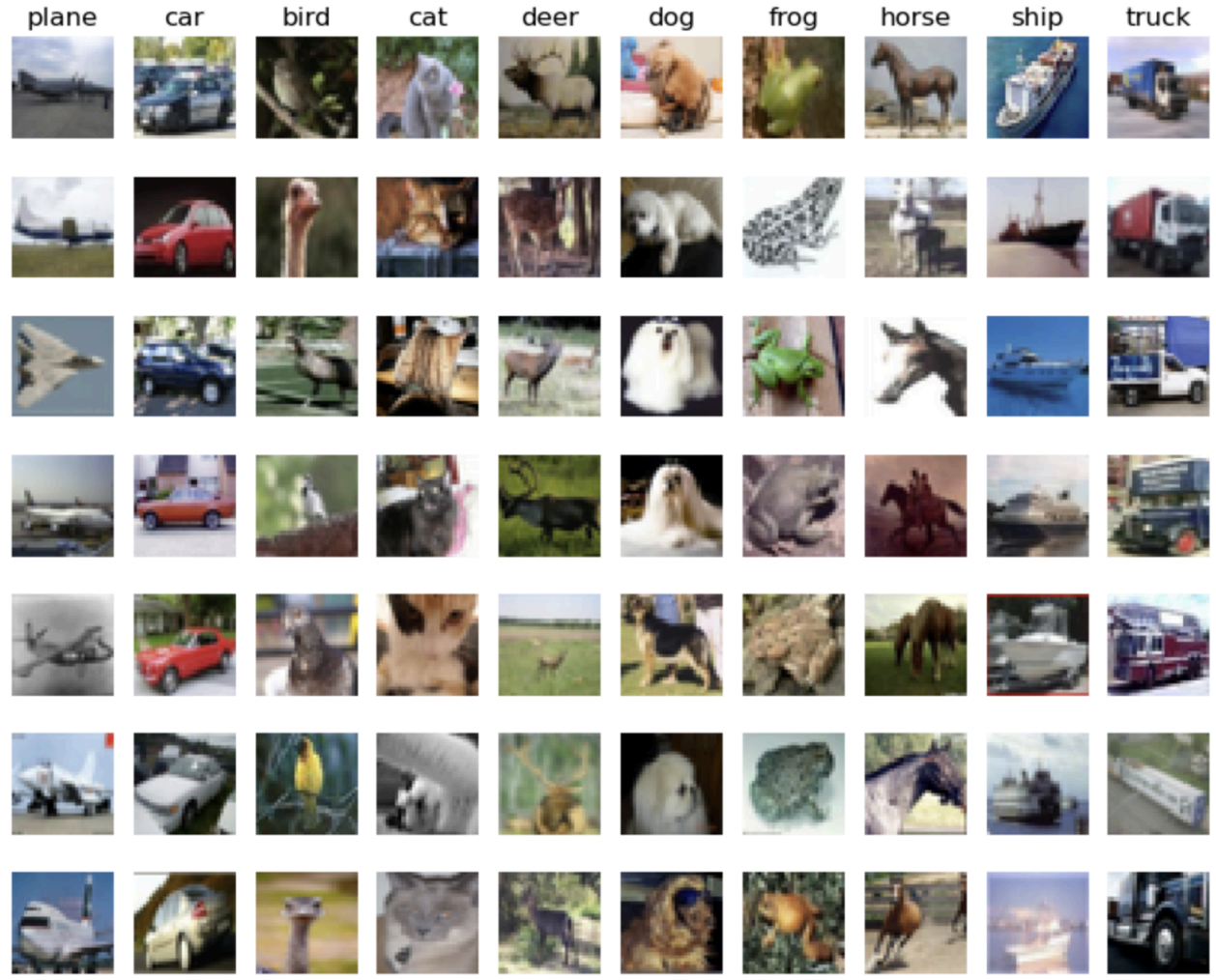
补全 `svm_loss_vectorized`：

```
scores = X.dot(W)
correct_scores = scores[np.arange(num_train), y].reshape(-1, 1)
margins = np.maximum(0, scores - correct_scores + 1)
margins[np.arange(num_train), y] = 0
loss = np.sum(margins) / num_train + reg * np.sum(W * W)

mask = (margins > 0).astype(float)
row_sum = np.sum(mask, axis=1)
mask[np.arange(num_train), y] = -row_sum
dW = X.T.dot(mask) / num_train + 2 * reg * W
```

说明：用 margin 矩阵一次性计算损失；mask 统计每行正 margin 个数并修正正确类梯度，最终平均并加正则。

### 3, ipynb 相关检查结果

- 数值梯度检查相对误差 ~1e-6 至 1e-4，满足要求。
- 朴素与向量化损失一致：Naive/Vectorized loss = 9.041890；向量化速度约快千倍。
- 训练/测试集形状、展平/加偏置后维度与日志一致。

- 


- ```
  Train data shape:  (49000, 32, 32, 3)
  Train labels shape:  (49000,)
  Validation data shape:  (1000, 32, 32, 3)
  Validation labels shape:  (1000,)
  Test data shape:  (1000, 32, 32, 3)
  Test labels shape:  (1000,)
  ```

  Generate  Code  Markdown

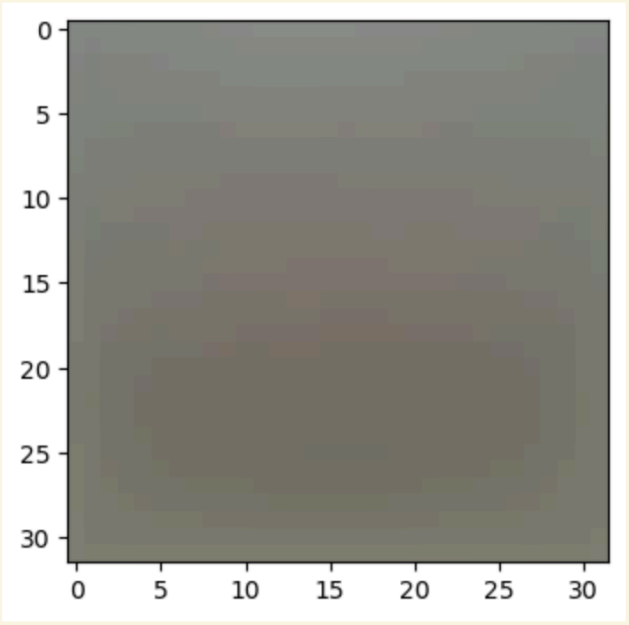
  Add Code Cell

  ```python
  # 预处理，将图像数据变成向量
  X_train = np.reshape(X_train, (X_train.shape[0], -1))
  X_val = np.reshape(X_val, (X_val.shape[0], -1))
  X_test = np.reshape(X_test, (X_test.shape[0], -1))
  X_dev = np.reshape(X_dev, (X_dev.shape[0], -1))

  # As a sanity check, print out the shapes of the data
  print('Training data shape: ', X_train.shape)
  print('Validation data shape: ', X_val.shape)
  print('Test data shape: ', X_test.shape)
  print('dev data shape: ', X_dev.shape)
  ```

  [6]  × pdf-ignore  + Tag  Python

  ```
  Training data shape:  (49000, 3072)
  Validation data shape:  (1000, 3072)
  Test data shape:  (1000, 3072)
  dev data shape:  (4000, 3072)
  ```
  ```

- 


```
(49000, 3073) (1000, 3073) (1000, 3073) (4000, 3073)
```

## 二、实现SGD （10%）

`linear_classifier.py` 关键代码：

```
# 采样 mini-batch
indices = np.random.choice(num_train, batch_size, replace=True)
X_batch, y_batch = X[indices], y[indices]
# 梯度更新
loss, grad = self.loss(X_batch, y_batch, reg)
self.W -= learning_rate * grad
```

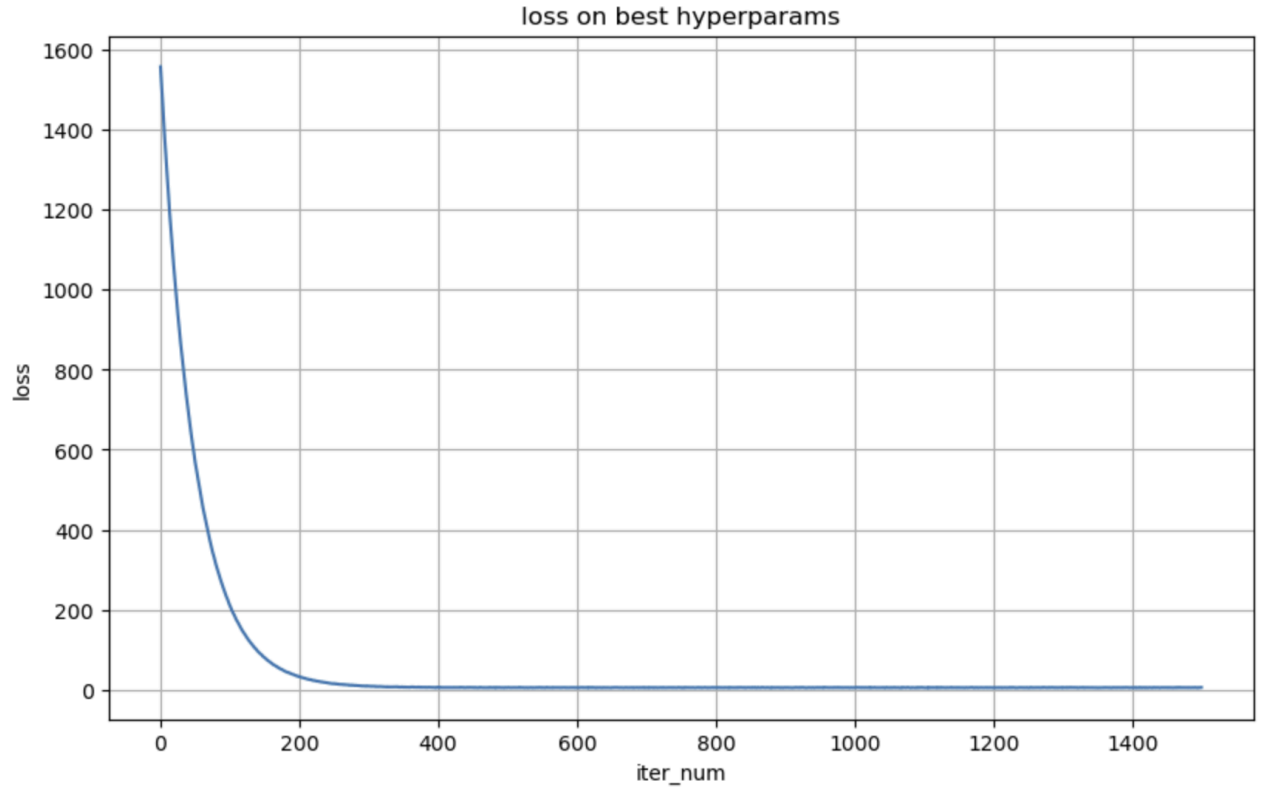
说明：有放回随机采样，按梯度下降更新权重。

## 三、利用验证集做超参数调优 （10%）

网格 (5×5)：learning_rate ∈ [1e-8, 5e-8, 1e-7, 5e-7, 1e-6]，reg ∈ [1e4, 2.5e4, 5e4, 1e5, 2.5e5]。验证集准确率表：

| lr \ reg | 1e4 | 2.5e4 | 5e4 | 1e5 | 2.5e5 |
|---|---|---|---|---|---|
| 1e-8 | 0.204 | 0.187 | 0.164 | 0.192 | 0.270 |
| 5e-8 | 0.265 | 0.259 | 0.348 | 0.349 | 0.329 |
| 1e-7 | 0.285 | 0.346 | **0.363** | 0.351 | 0.303 |
| 5e-7 | 0.342 | 0.347 | 0.293 | 0.345 | 0.247 |
| 1e-6 | 0.311 | 0.254 | 0.242 | 0.245 | 0.218 |

- 最优：lr = 1e-7，reg = 5e4，验证集 acc ≈ 0.363。

- 用最佳超参重训 1500 iter 后：accuracy ≈ 0.343（日志输出）。
- 


## 训练精度参考

- 基础 SVM 训练（示例 lr=1e-7, reg=2.5e4, 1500 iter）：train acc ≈ 0.360，test acc ≈ 0.368。