图像分类
Image Classification

人工智能学院

人工智能与网络搜索教研中心

胡佳妮 jnhu@bupt.edu.cn

内容提要

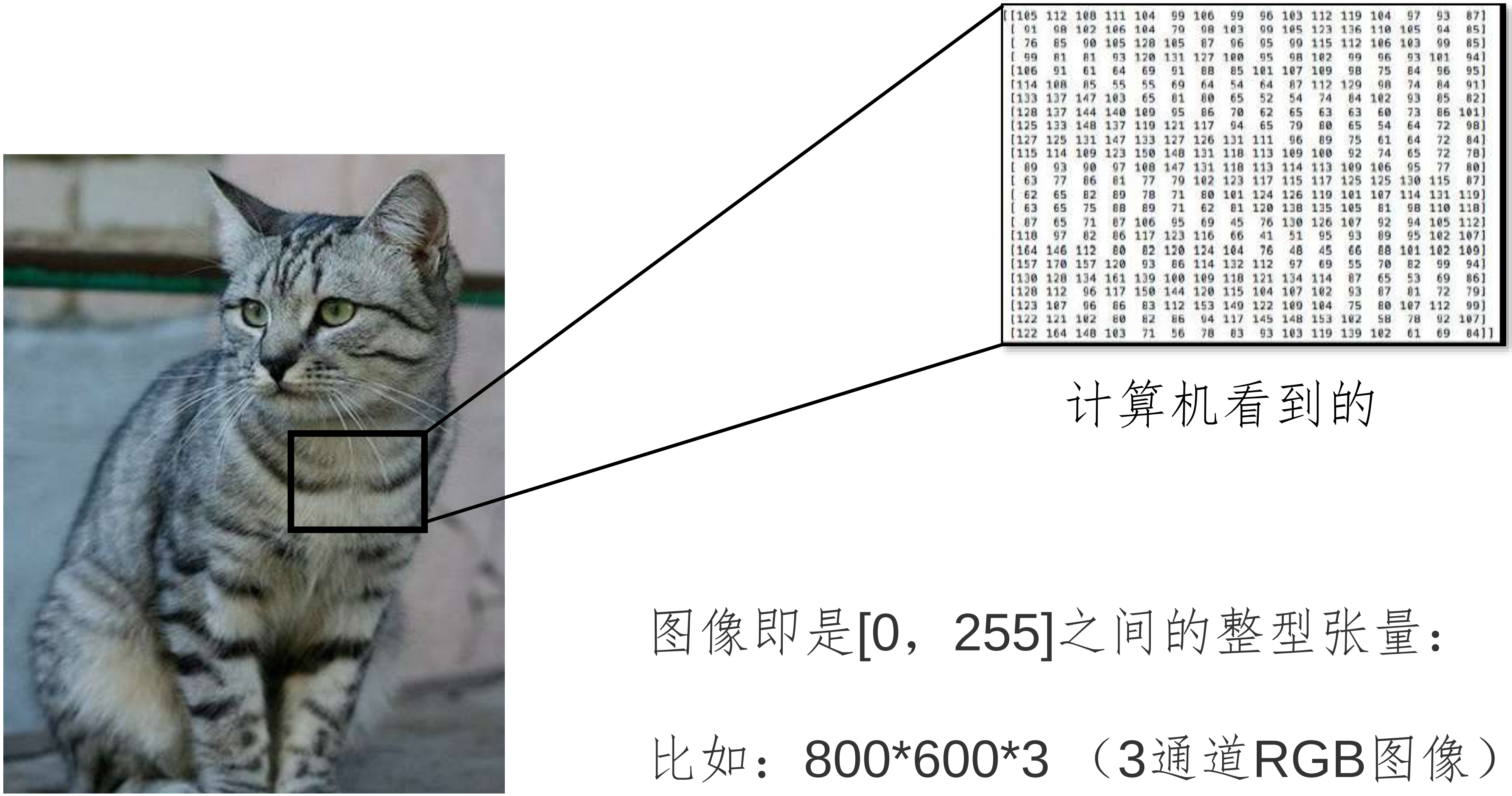
- K近邻算法
- 线性分类器：SVM、Softmax
- 两层神经元网络
- 图像特征

图像分类：计算机视觉的核心任务

{假设给定一组标签}
{dog, cat, truck, plane, …}

cat

问题：语义鸿沟

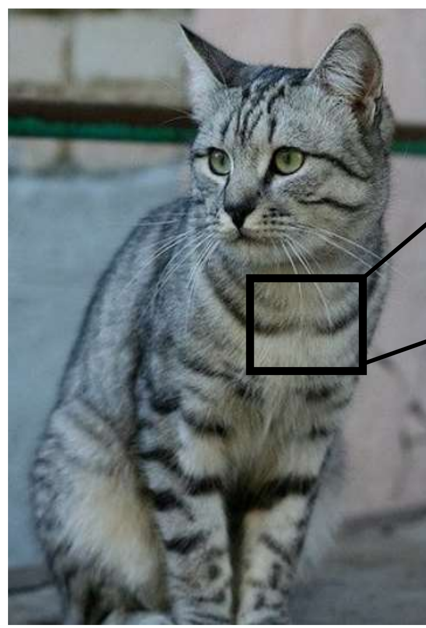


计算机看到的

图像即是[0，255]之间的整型张量：

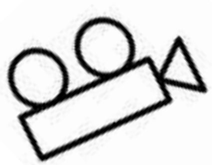
比如：800*600*3 （3通道RGB图像）

挑战：视角变化



```
[[105 112 108 111 104  99 106  99  96 103 112 119 104  97  93  87]
 [ 91  98 102 106 104  79  98 103  99 105 123 136 110 105  94  85]
 [ 76  85  90 105 128 105  87  96  95  99 115 112 106 103  99  85]
 [ 99  81  81  93 120 131 127 100  95  98 102  99  96  93 101  94]
 [106  91  61  64  69  91  88  85 101 107 109  98  75  84  96  95]
 [114 108  85  55  55  69  64  54  64  87 112 129  98  74  84  91]
 [133 137 147 103  65  81  80  65  52  54  74  84 102  93  85  82]
 [128 137 144 140 109  95  86  70  62  65  63  63  60  73  86 101]
 [125 133 148 137 119 121 117  94  65  79  80  65  54  64  72  98]
 [127 125 131 147 133 127 126 131 111  96  89  75  61  64  72  84]
 [115 114 109 123 150 148 131 118 113 109 100  92  74  65  72  78]
 [ 89  93  90  97 108 147 131 118 113 114 113 109 106  95  77  80]
 [ 63  77  86  81  77  79 102 123 117 115 117 125 125 130 115  87]
 [ 62  65  82  89  78  71  80 101 124 126 119 101 107 114 131 119]
 [ 63  65  75  88  89  71  62  81 120 138 135 105  81  98 110 118]
 [ 87  65  71  87 106  95  69  45  76 130 126 107  92  94 105 112]
 [118  97  82  86 117 123 116  66  41  51  95  93  89  95 102 107]
 [164 146 112  80  82 120 124 104  76  48  45  66  88 101 102 109]
 [157 170 157 120  93  86 114 132 112  97  69  55  70  82  99  94]
 [130 128 134 161 139 100 109 118 121 134 114  87  65  53  69  86]
 [128 112  96 117 150 144 120 115 104 107 102  93  87  81  72  79]
 [123 107  96  86  83 112 153 149 122 109 104  75  80 107 112  99]
 [122 121 102  80  82  86  94 117 145 148 153 102  58  78  92 107]
 [122 164 148 103  71  56  78  83  93 103 119 139 102  61  69  84]]
```

当相机移动时，所
有像素值都会改变！



挑战：背景干扰

挑战：光照

挑战：遮挡



挑战：形变

挑战：类内差异

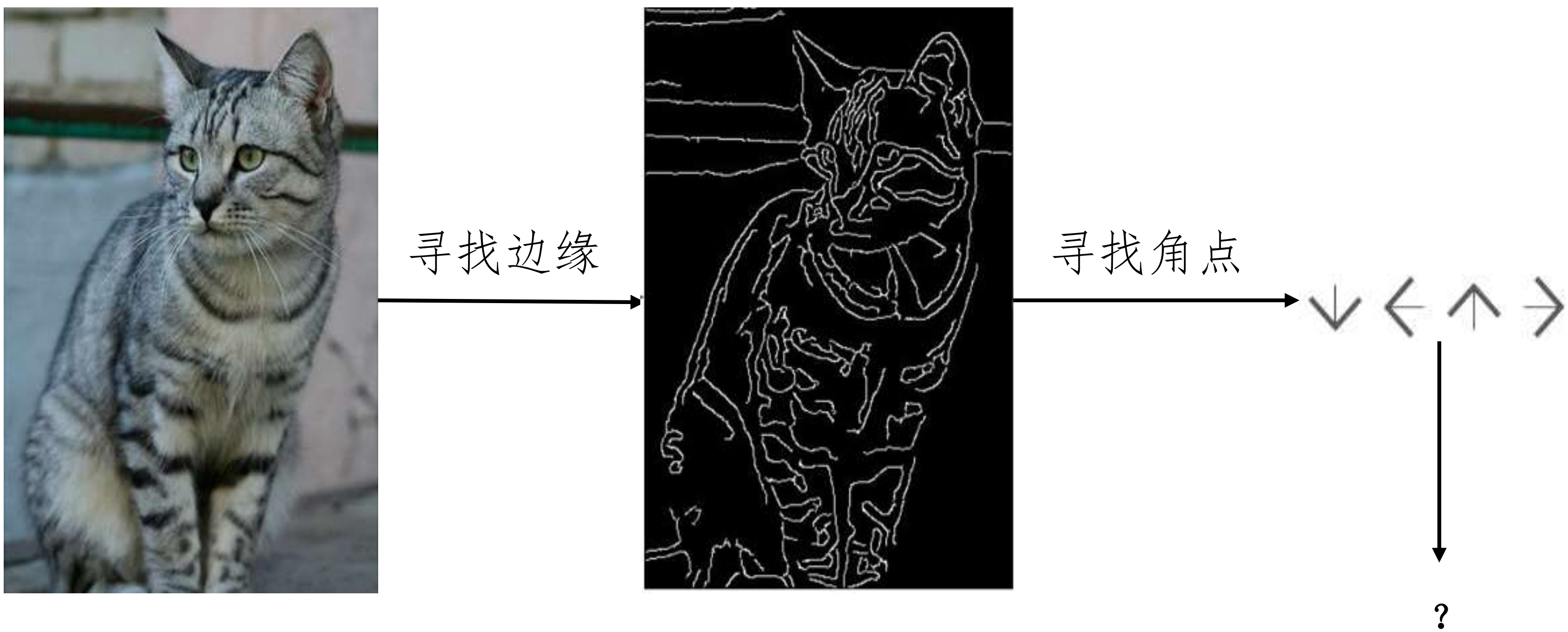
图像分类器

```
def classify_image(image):
  # Some magic here?
  return class_label
```

与对一列数字进行排序不同

没有显而易见的方式对算法进行硬编码，来识别猫，或其他类

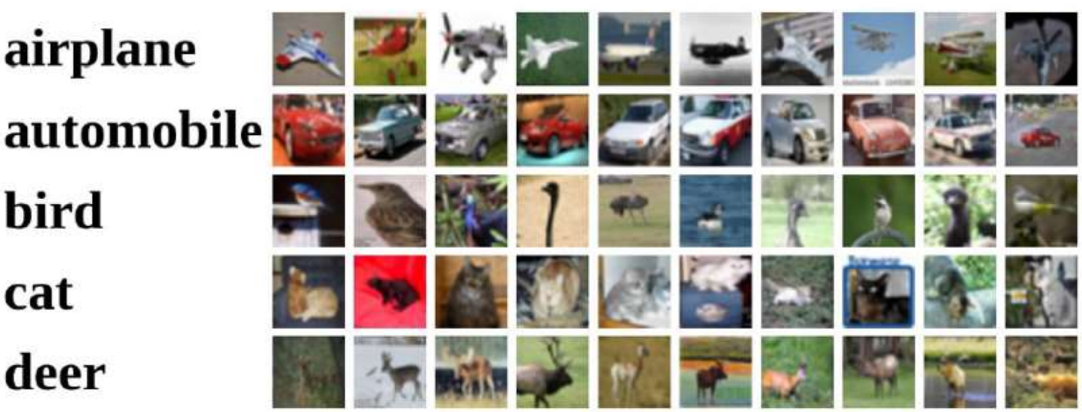
做出的尝试



机器学习：数据驱动方法

1. 收集图像和标签的数据集
2. 使用机器学习算法训练分类器
3. 在新图像上评估分类器

训练集样例




```python
def train(images, labels):
  # Machine learning!
  return model
```

```python
def predict(model, test_images):
  # Use model to predict labels
  return test_labels
```

K-近邻算法

最基础分类器：**最近邻**

```
def train(images, labels):
  # Machine learning!
  return model
```

记忆所有的数据与标签

```
def predict(model, test_images):
  # Use model to predict labels
  return test_labels
```

预测最相似训练图像的标签

第一分类器：**最近邻**

deer bird plane cat car

带标签的训练数据

?

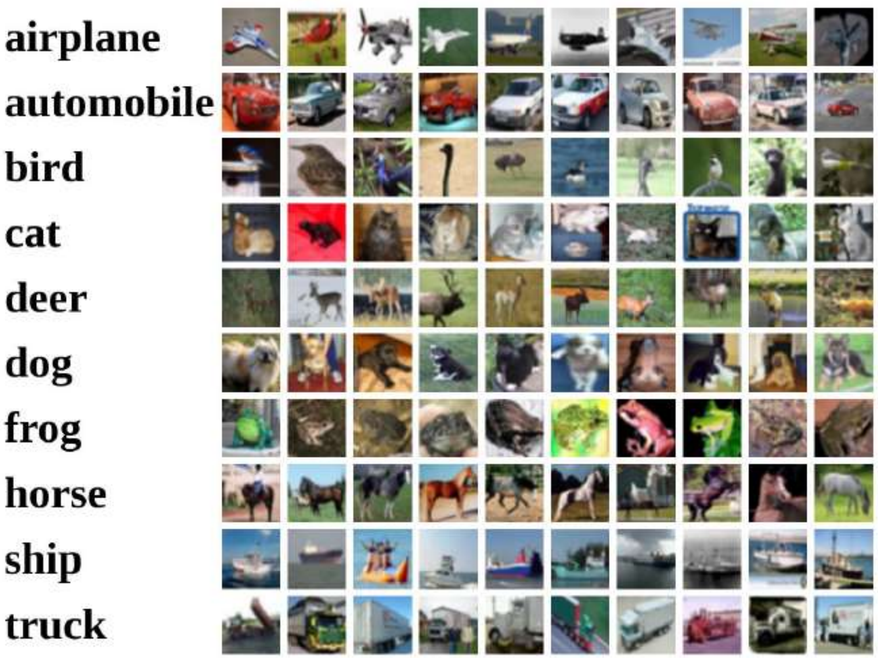
查询数据

距离度量

|  ,  | → $\mathbb{R}$

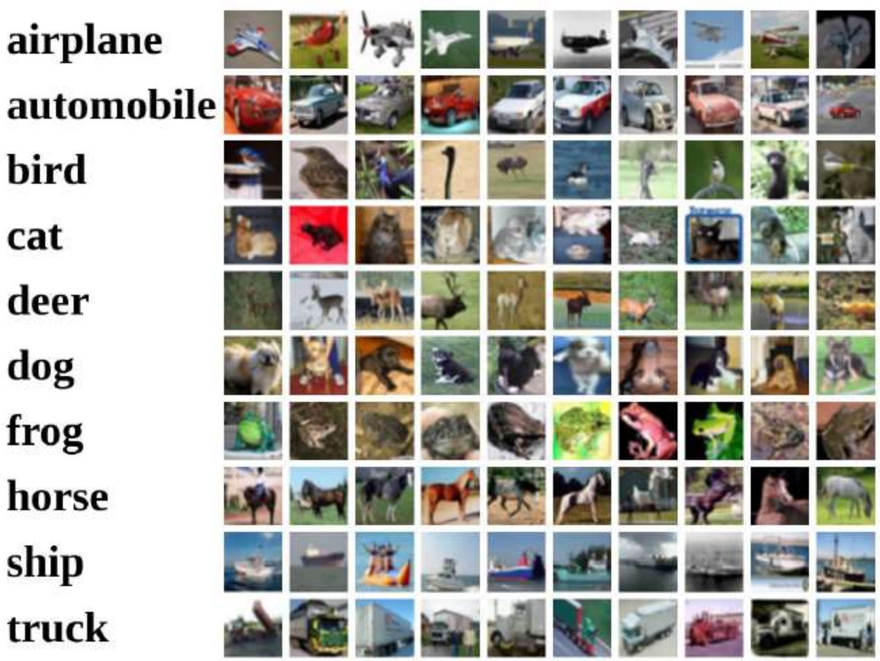
数据库样例：CIFAR10

10 类
50,000 训练图像
10,000 测试图像



数据库样例：**CIFAR10**

10 类
50,000 训练图像
10,000 测试图像



测试图像与最近邻

距离度量：用于比较图像

L_1距离：$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$

测试图像 训练图像 像素差绝对值

$$\left| \begin{bmatrix} 56 & 32 & 10 & 18 \\ 90 & 23 & 128 & 133 \\ 24 & 26 & 178 & 200 \\ 2 & 0 & 255 & 220 \end{bmatrix} - \begin{bmatrix} 10 & 20 & 24 & 17 \\ 8 & 10 & 89 & 100 \\ 12 & 16 & 178 & 170 \\ 4 & 32 & 233 & 112 \end{bmatrix} \right| = \begin{bmatrix} 46 & 12 & 14 & 1 \\ 82 & 13 & 39 & 33 \\ 12 & 10 & 0 & 30 \\ 2 & 32 & 22 & 108 \end{bmatrix} \xrightarrow{\text{相加}} 456$$

最近邻分类器

```
import numpy as np

class NearestNeighbor:
  def __init__(self):
    pass

  def train(self, X, y):
    """ X is N x D where each row is an example. Y is 1-dimension of size N """
    # the nearest neighbor classifier simply remembers all the training data
    self.Xtr = X
    self.ytr = y

  def predict(self, X):
    """ X is N x D where each row is an example we wish to predict label for """
    num_test = X.shape[0]
    # lets make sure that the output type matches the input type
    Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

    # loop over all test rows
    for i in xrange(num_test):
      # find the nearest training image to the i'th test image
      # using the L1 distance (sum of absolute value differences)
      distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
      min_index = np.argmin(distances) # get the index with smallest distance
      Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

    return Ypred
```

最近邻分类器

```python
import numpy as np

class NearestNeighbor:
  def __init__(self):
    pass

  def train(self, X, y):
    """ X is N x D where each row is an example. Y is 1-dimension of size N """
    # the nearest neighbor classifier simply remembers all the training data
    self.Xtr = X
    self.ytr = y

  def predict(self, X):
    """ X is N x D where each row is an example we wish to predict label for """
    num_test = X.shape[0]
    # lets make sure that the output type matches the input type
    Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

    # loop over all test rows
    for i in xrange(num_test):
      # find the nearest training image to the i'th test image
      # using the L1 distance (sum of absolute value differences)
      distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
      min_index = np.argmin(distances) # get the index with smallest distance
      Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

    return Ypred
```

记忆训练数据

```python
import numpy as np

class NearestNeighbor:
  def __init__(self):
    pass

  def train(self, X, y):
    """ X is N x D where each row is an example. Y is 1-dimension of size N """
    # the nearest neighbor classifier simply remembers all the training data
    self.Xtr = X
    self.ytr = y

  def predict(self, X):
    """ X is N x D where each row is an example we wish to predict label for """
    num_test = X.shape[0]
    # lets make sure that the output type matches the input type
    Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

    # loop over all test rows
    for i in xrange(num_test):
      # find the nearest training image to the i'th test image
      # using the L1 distance (sum of absolute value differences)
      distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
      min_index = np.argmin(distances) # get the index with smallest distance
      Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

    return Ypred
```

最近邻分类器

对每一个测试图像：
寻找最近的训练图像
预测最近图像的标签

```python
import numpy as np

class NearestNeighbor:
  def __init__(self):
    pass

  def train(self, X, y):
    """ X is N x D where each row is an example. Y is 1-dimension of size N """
    # the nearest neighbor classifier simply remembers all the training data
    self.Xtr = X
    self.ytr = y

  def predict(self, X):
    """ X is N x D where each row is an example we wish to predict label for """
    num_test = X.shape[0]
    # lets make sure that the output type matches the input type
    Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

    # loop over all test rows
    for i in xrange(num_test):
      # find the nearest training image to the i'th test image
      # using the L1 distance (sum of absolute value differences)
      distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
      min_index = np.argmin(distances) # get the index with smallest distance
      Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

    return Ypred
```

最近邻分类器

Q： 对于N个样例，训练和预测速度各为多少？

A. 训练：O(1)
   预测：O(1)

B. 训练：O(1)
   预测：O(N)

C. 训练：O(N)
   预测：O(1)

D. 训练：O(N)
   预测：O(N)

```python
import numpy as np

class NearestNeighbor:
  def __init__(self):
    pass

  def train(self, X, y):
    """ X is N x D where each row is an example. Y is 1-dimension of size N """
    # the nearest neighbor classifier simply remembers all the training data
    self.Xtr = X
    self.ytr = y

  def predict(self, X):
    """ X is N x D where each row is an example we wish to predict label for """
    num_test = X.shape[0]
    # lets make sure that the output type matches the input type
    Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

    # loop over all test rows
    for i in xrange(num_test):
      # find the nearest training image to the i'th test image
      # using the L1 distance (sum of absolute value differences)
      distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
      min_index = np.argmin(distances) # get the index with smallest distance
      Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

    return Ypred
```

最近邻分类器

Q： 对于N个样例，训练和预测速度各为多少？

A. 训练：O(1)
   预测：O(1)

B. 训练：O(1)
   预测：O(N)

C. 训练：O(N)
   预测：O(1)

D. 训练：O(N)
   预测：O(N)

```python
import numpy as np

class NearestNeighbor:
  def __init__(self):
    pass

  def train(self, X, y):
    """ X is N x D where each row is an example. Y is 1-dimension of size N """
    # the nearest neighbor classifier simply remembers all the training data
    self.Xtr = X
    self.ytr = y

  def predict(self, X):
    """ X is N x D where each row is an example we wish to predict label for """
    num_test = X.shape[0]
    # lets make sure that the output type matches the input type
    Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

    # loop over all test rows
    for i in xrange(num_test):
      # find the nearest training image to the i'th test image
      # using the L1 distance (sum of absolute value differences)
      distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
      min_index = np.argmin(distances) # get the index with smallest distance
      Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

    return Ypred
```

最近邻分类器

Q： 对于N个样例，训练和预测速度各为多少？

ANS： 训练：O(1)，预测：O(N)

这样并不好，我们希望分类器可以在预测时速度**快**，而训练时速度**慢**是可以接受的。

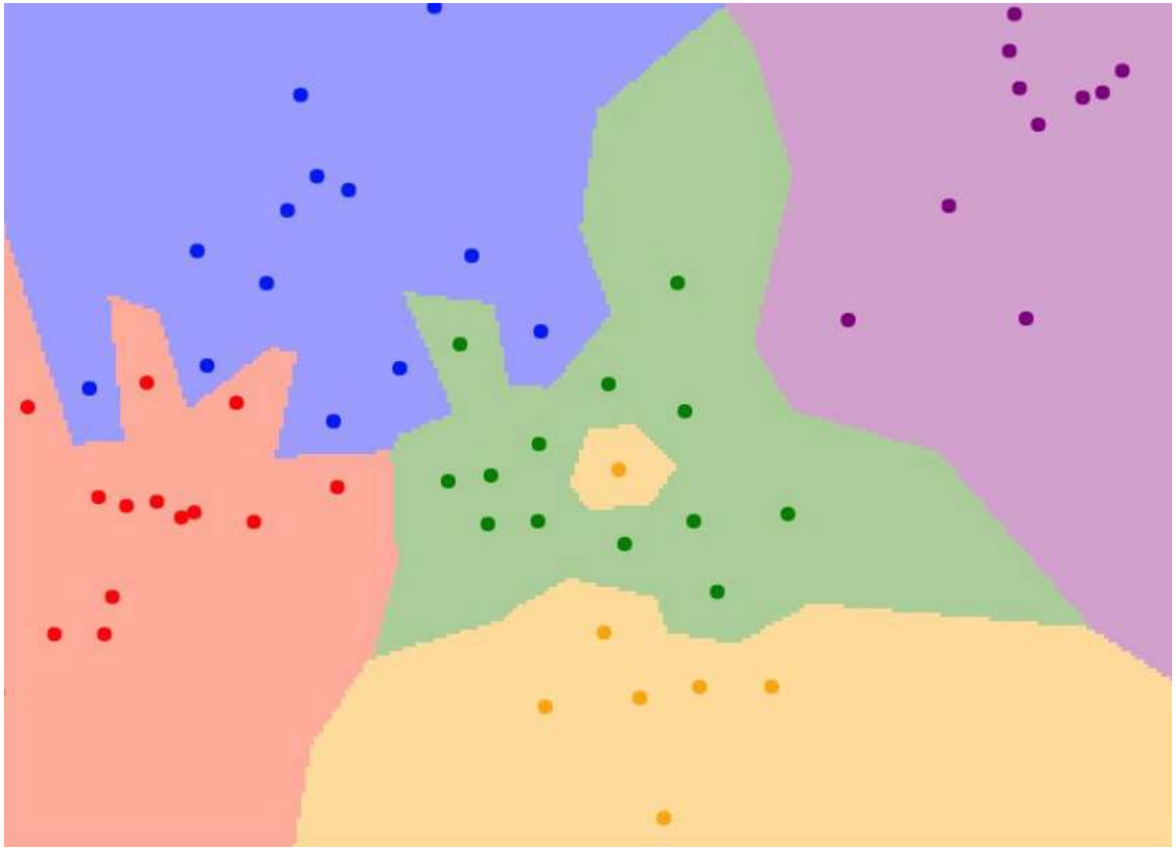
```python
import numpy as np

class NearestNeighbor:
  def __init__(self):
    pass

  def train(self, X, y):
    """ X is N x D where each row is an example. Y is 1-dimension of size N """
    # the nearest neighbor classifier simply remembers all the training data
    self.Xtr = X
    self.ytr = y

  def predict(self, X):
    """ X is N x D where each row is an example we wish to predict label for """
    num_test = X.shape[0]
    # lets make sure that the output type matches the input type
    Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

    # loop over all test rows
    for i in xrange(num_test):
      # find the nearest training image to the i'th test image
      # using the L1 distance (sum of absolute value differences)
      distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
      min_index = np.argmin(distances) # get the index with smallest distance
      Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

    return Ypred
```

最近邻分类器

有许多方法可用于快速/近似近邻

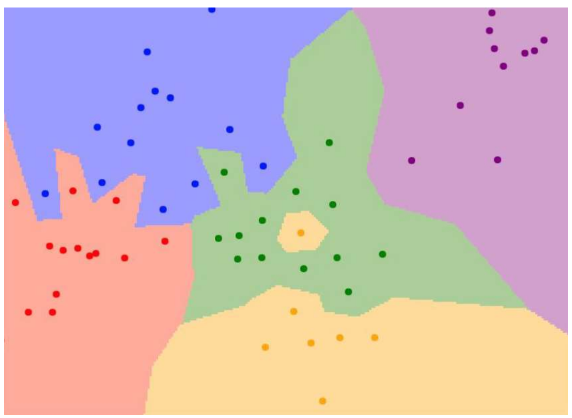
一个参考例子：https://github.com/facebookresearch/faiss

最近邻分类面



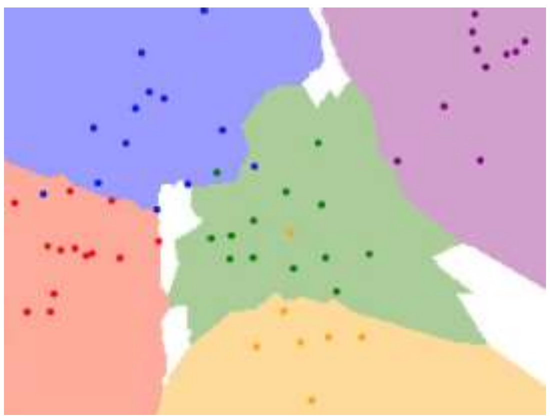
K-近邻

从K个最近的点进行**多数投票**，而不是从最近的邻居复制标签。



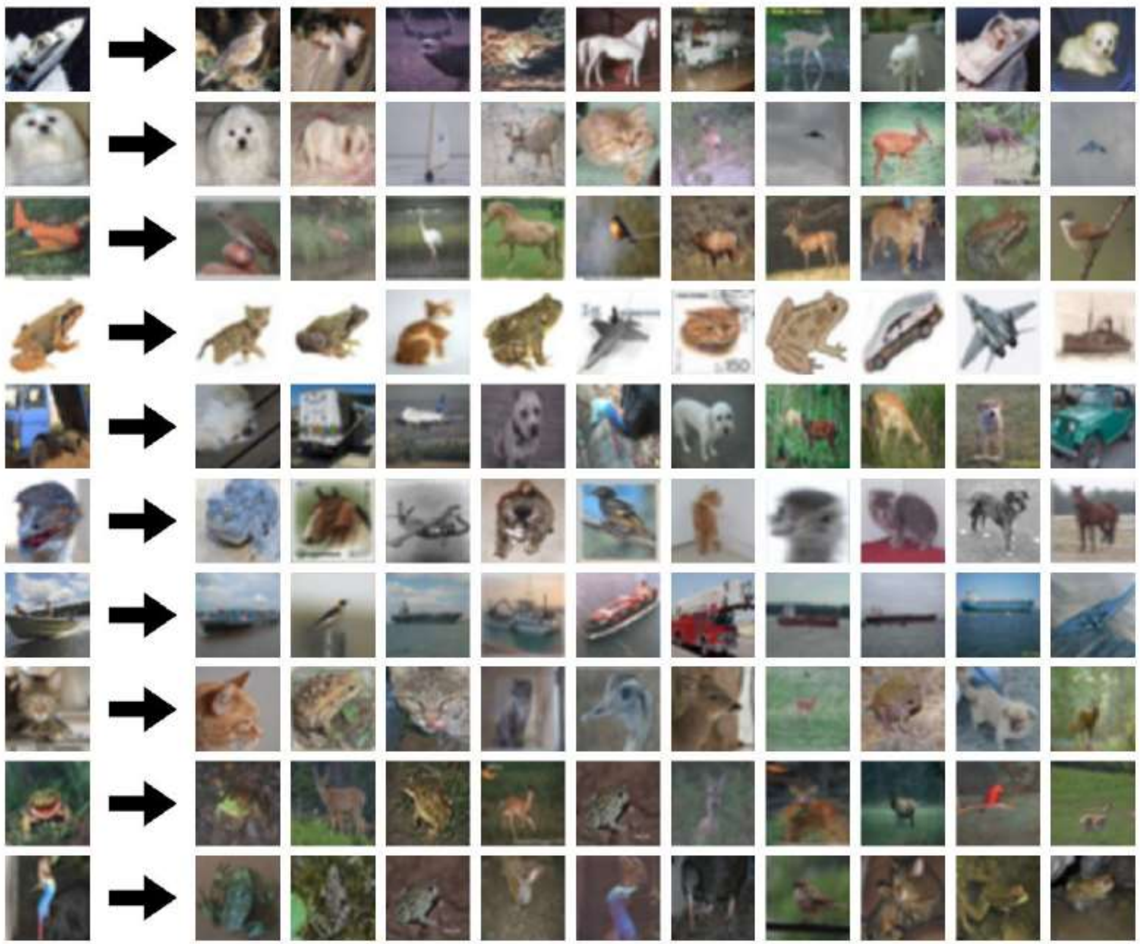
K=1

K=3



K=5

看起来像什么？

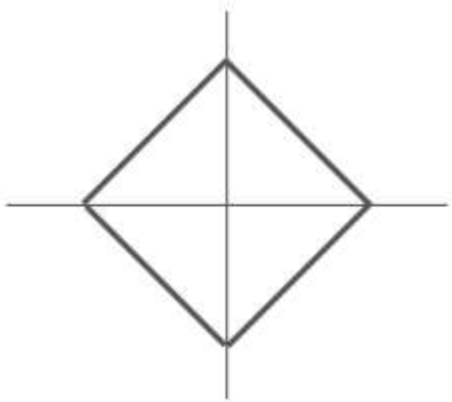


看起来像什么？

K-近邻：距离度量

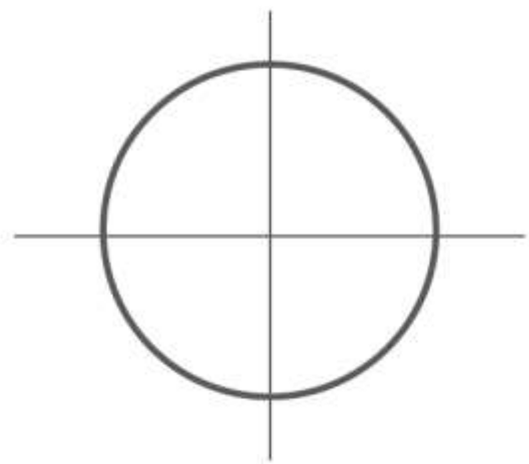
L_1（曼哈顿距离）距离

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$



L_2（欧式距离）距离

$$d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$$



K-近邻：距离度量

L_1（曼哈顿距离）距离

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$

K=1

L_2（欧式距离）距离

$$d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$$

K=1

超参数

k的最佳值是多少？

最佳**距离**是多少？

这些均是超参数：关于算法本身的选择。

超参数

k的最佳值是多少？

最佳**距离**是多少？

这些均是超参数：关于算法本身的选择。

非常依赖问题。
一定要把它们都试一下，看看什么最有效。

设置超参数

想法1：选择一个对数据集运行最佳的超参数

数据集

设置超参数

想法1：选择一个对数据集运行最佳的超参数

不好：K=1 总是在训练集上运行的最好

数据集

设置超参数

想法1：选择一个对数据集运行最佳的超参数

不好：K=1 总是在训练集上运行的最好

数据集

想法2：将数据分为**训练数据**和**测试数据**，选择对测试数据最有效的超参数

训练 测试

设置超参数

想法1：选择一个对数据集运行最佳的超参数

不好：K=1 总是在训练集上运行的最好

数据集

想法2：将数据分为**训练数据**和**测试数据**，选择对测试数据最有效的超参数

不好：不知道算法将如何执行新数据

训练 | **测试**

设置超参数

想法1：选择一个对数据集运行最佳的超参数

不好：K=1 总是在训练集上运行的最好

数据集

想法2：将数据分为**训练数据**和**测试数据**，选择对测试集最有效的超参数

不好：不知道算法将如何执行新数据

训练	测试

想法3：将数据分为**训练集**、**验证集**和**测试集**，选择对验证集最有效的超参数，并在测试集上

好多了！

训练	验证	测试

设置超参数

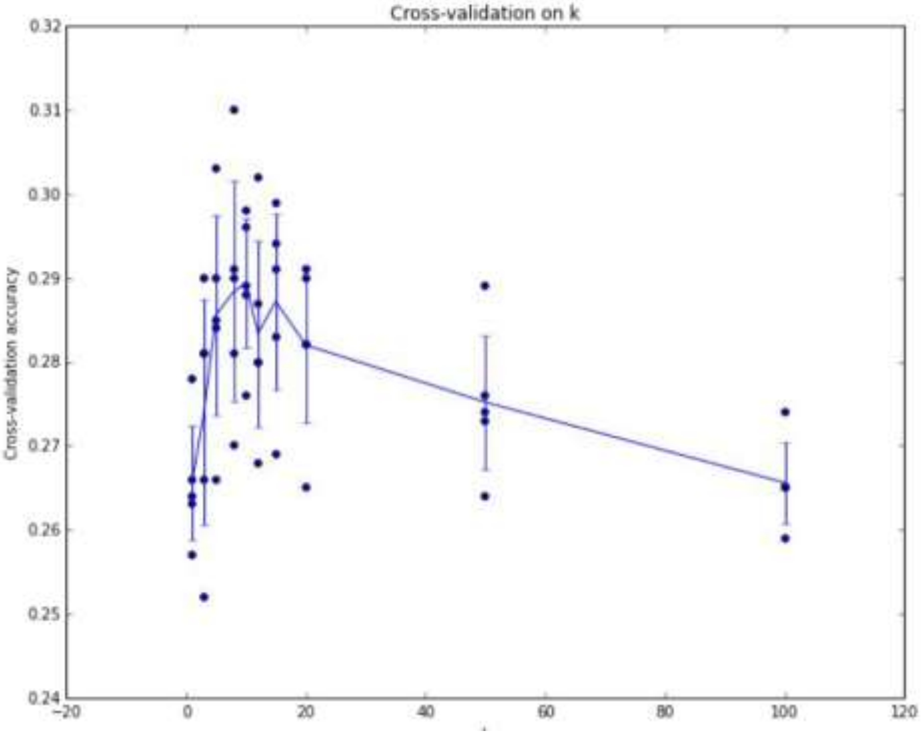
数据集

想法4：交叉验证：将数据分成多个部分，每个部分作为验证进行尝试，并平均结果

包 1	包 2	包 3	包 4	包 5	测试集
包 1	包 2	包 3	包 4	包 5	测试集
包 1	包 2	包 3	包 4	包 5	测试集

对于小数据集很有用，但在深度学习中使用频率不太高

设置超参数



示例：横轴为**k**的5-折交叉验证

每一点表示一个结果

曲线穿过平均值，竖条表示标准差

（k~=7时，最适合此数据集）

像素距离上的k-近邻**从未被使用过**

-像素上的距离度量没有足够多的信息
-测试时非常慢

原图

局部遮挡

平移

变色

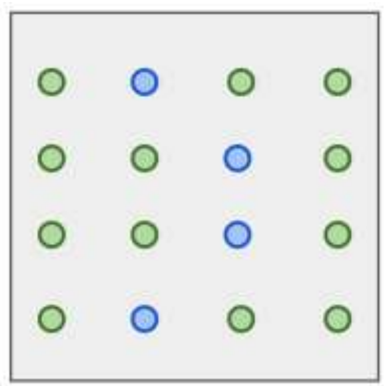
（所有3个图像与左侧图像的L_2距离相同）

像素距离的k-近邻**从未被使用过**

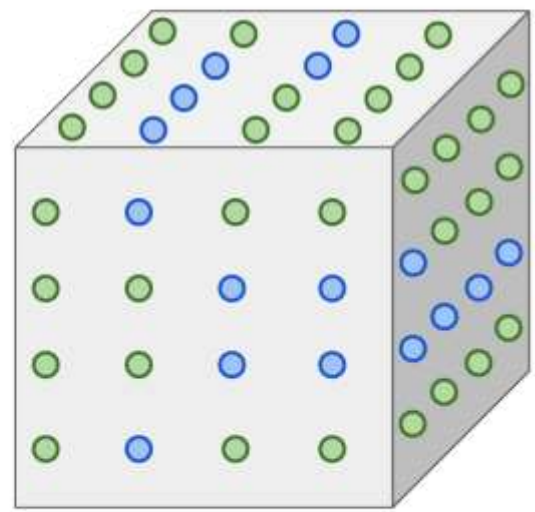
-维数灾难

维度 = 1
点数 = 4

维度 = 2
点数 = 4^2



维度 = 3
点数 = 4^3



K-近邻：总结

在**图像分类**中，我们从图像和标签的**训练集**开始，并且必须预测**测试集**中的标签

K-近邻分类器基于K-近邻训练样本预测标签

距离度量和K是**超参数**

使用**验证集**选择超参数

最后只在测试集上运行

像素距离不是很有用

线性分类器

参数法

图像

32×32×3的数组
（共3072个数）

$$f(x, W)$$

10个类别
评分数字

W

参数或权重

参数法：线性分类器

$$f(x, W) = Wx$$

图像

32×32×3的数组
（共3072个数）

$$f(\color{cyan}{x}, \color{red}{W})$$

10个类别
评分数字

$\color{red}{\mathbf{W}}$

参数或权重

参数法：线性分类器

3072×1

$$\boxed{f(x,W)} = \boxed{W}\boxed{x}$$

10×1　　10×3072

图像

32×32×3的数组
（共3072个数）

$$f(x, W)$$

10个类别
评分数字

W

参数或权重

参数法：线性分类器

$$\boxed{f(x,W)} = \boxed{W}\boxed{x} + \boxed{b}$$

3072×1

10×1　　10×3072　10×1

图像

32×32×3的数组
（共3072个数）

$f(\mathbf{x},\mathbf{W})$ →　10个类别评分数字

W

参数或权重

神经网络

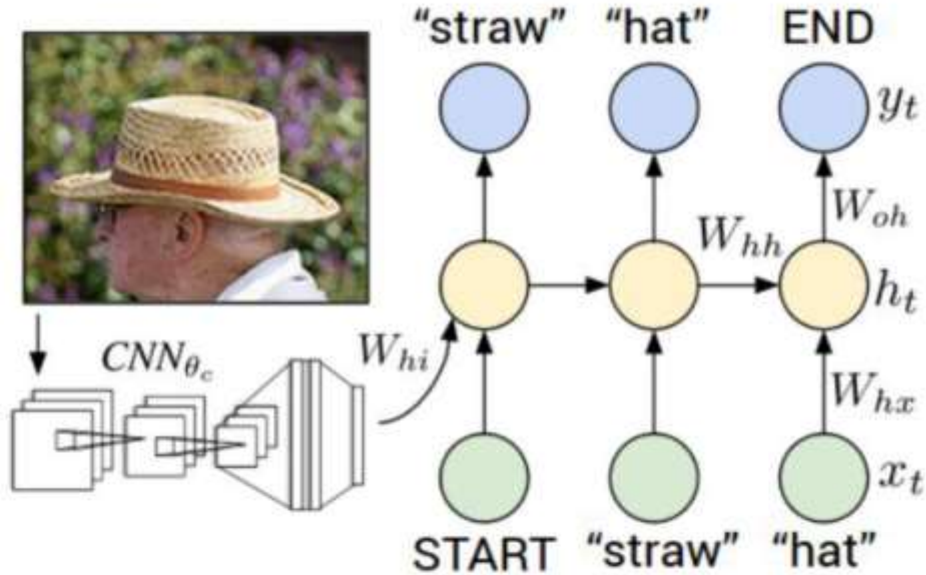
线性分类器

两个小女孩
在玩乐高

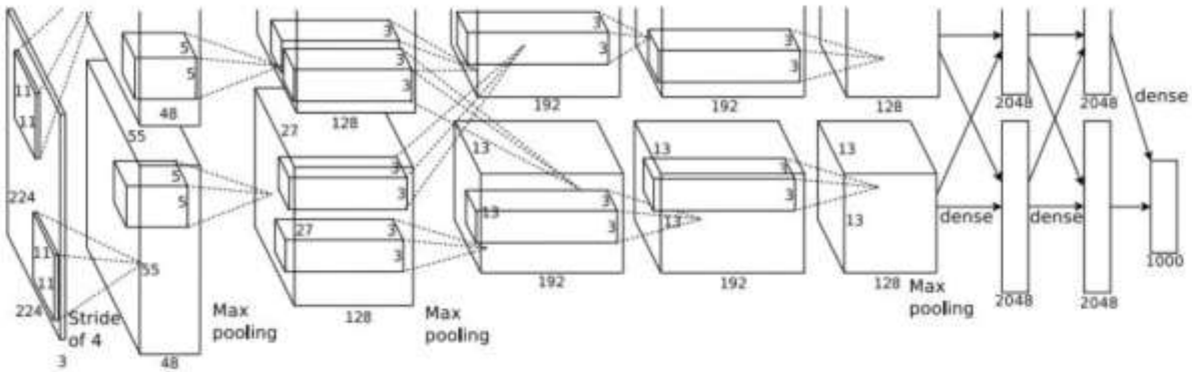
男孩在尾水板
上做后空翻

黑色短袖的男
人在弹吉他

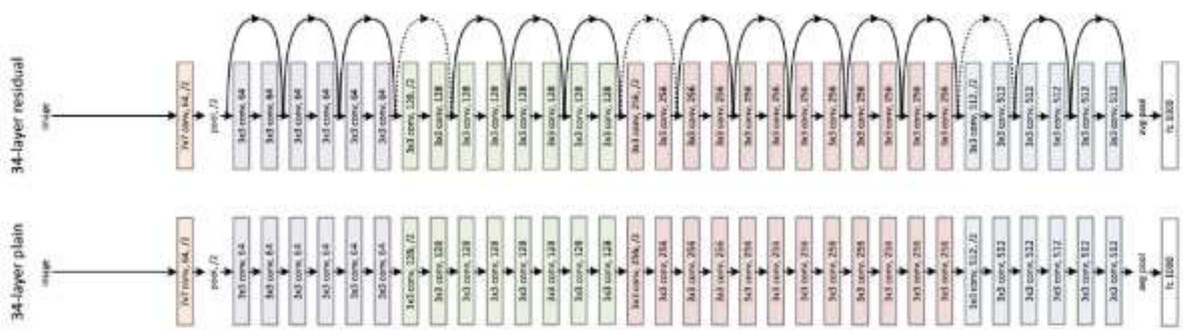
穿着橙色安全马
甲的建筑工人在
马路上工作



Karpathy and Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions", CVPR 2015



[Krizhevsky et al. 2012]



[He et al. 2015]

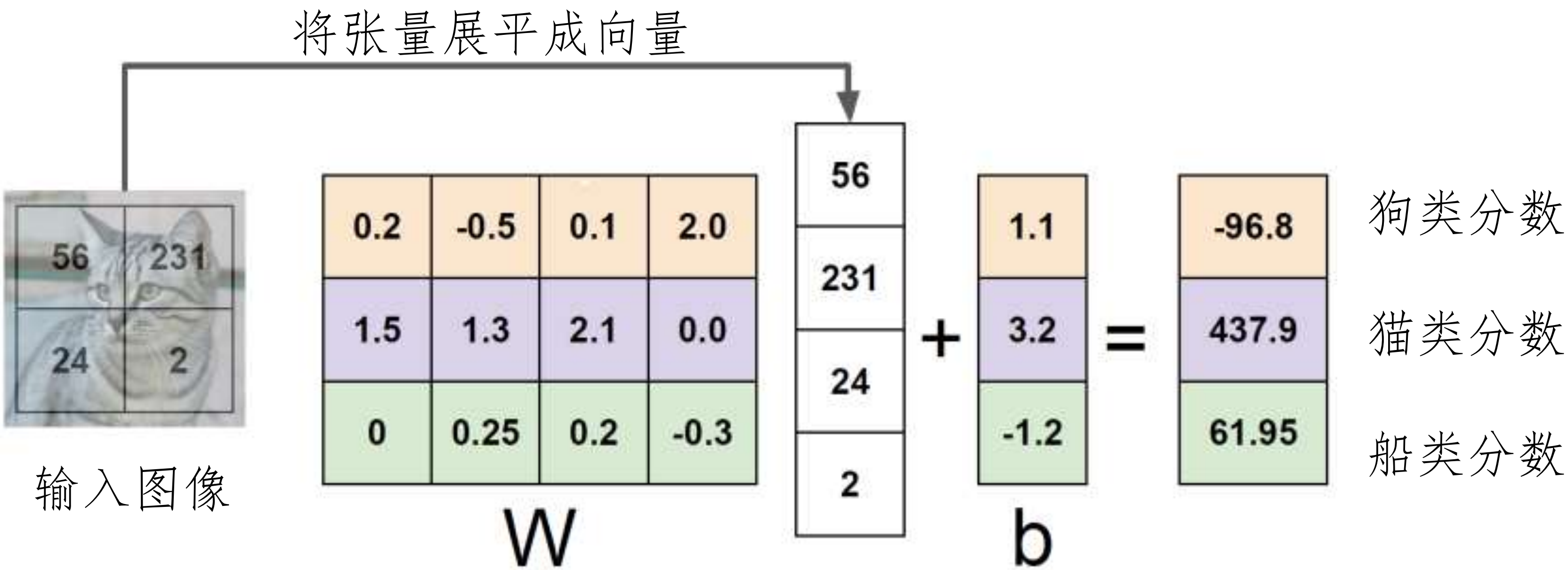
CIFAR10 图像分类

训练图像50,000张
每张图像为32x32x3

10,000张测试图像

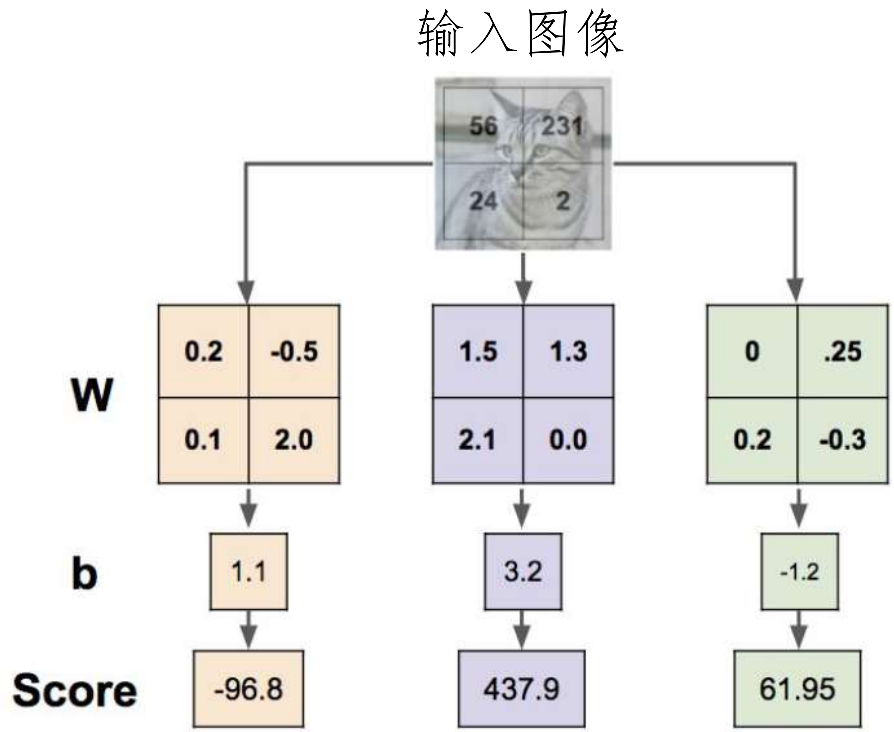
示例：一张有4个像素和3个类（猫/狗/船）的图片

示例：一张有4个像素和3个类（猫/狗/船）的图片



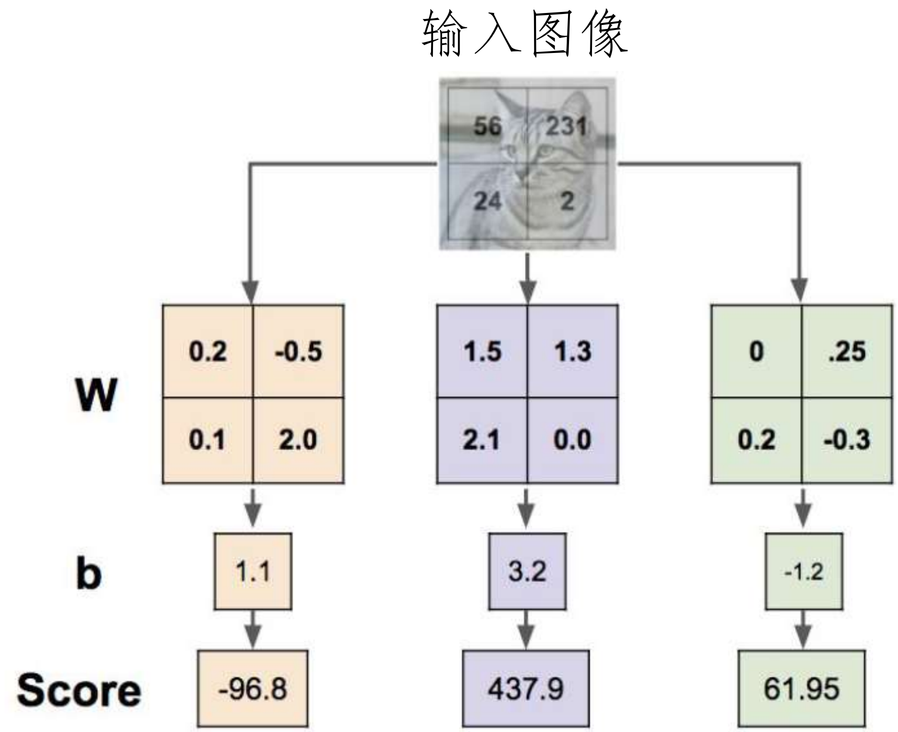
解释线性分类器


airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck


输入图像
56
231
24
2
W
0.2
-0.5
0.1
2.0
1.5
1.3
2.1
0.0
0
.25
0.2
-0.3
b
1.1
3.2
-1.2
Score
-96.8
437.9
61.95

解释线性分类器


airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck

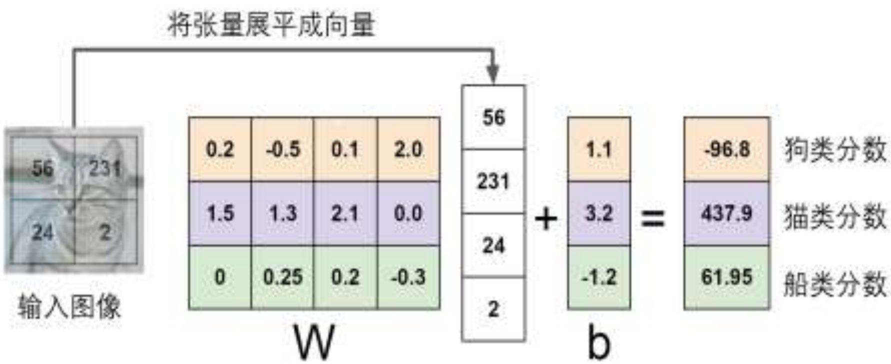

输入图像
56
231
24
2
W
0.2
-0.5
0.1
2.0
1.5
1.3
2.1
0.0
0
.25
0.2
-0.3
b
1.1
3.2
-1.2
Score
-96.8
437.9
61.95


plane
car
bird
cat
deer
dog
frog
horse
ship
truck

示例：一张有4个像素和3个类（猫/狗/船）的图片

<u>代数视角</u>

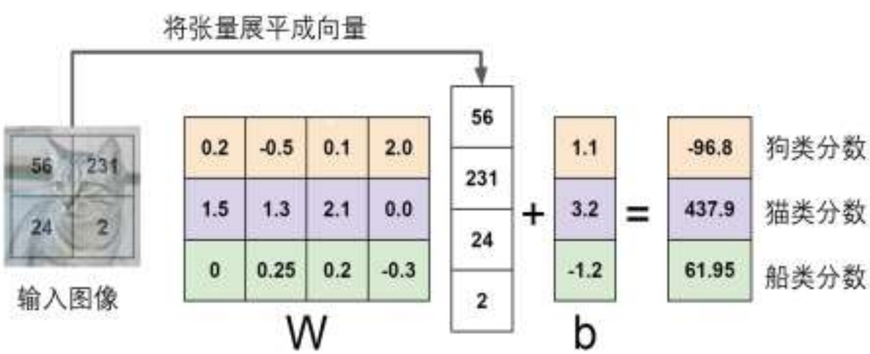
$$f(x,W)=Wx+b$$



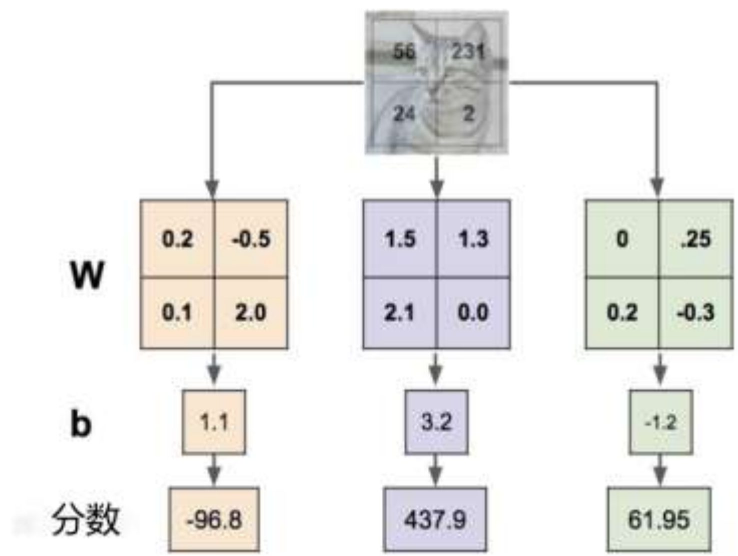
示例：一张有4个像素和3个类（猫/狗/船）的图片

代数视角

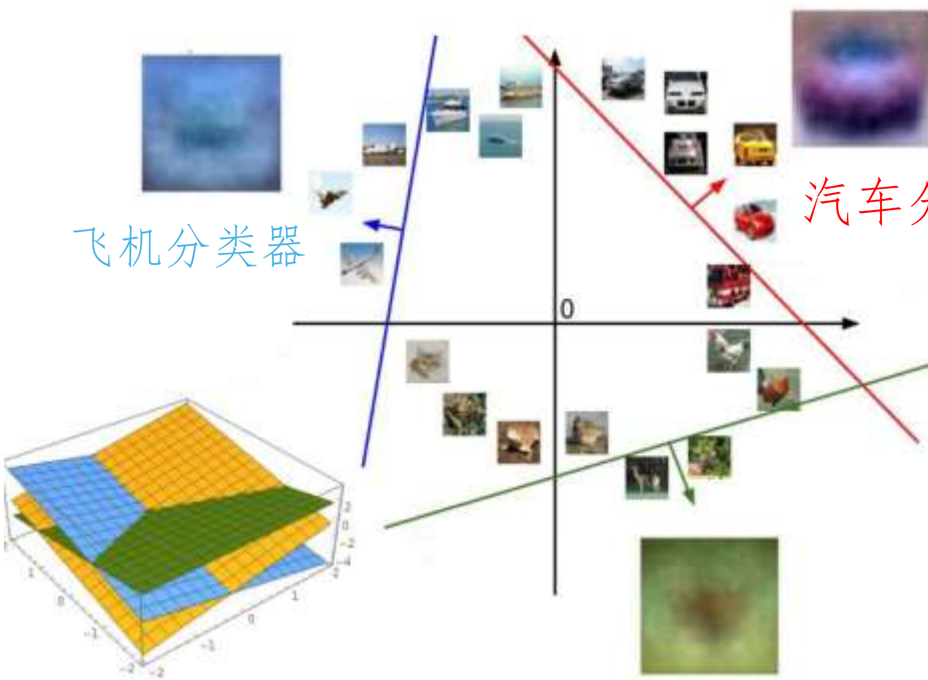
f(x,W)=Wx



视觉视角



解释线性分类器：几何视角



飞机分类器

汽车分类器

鹿分类器

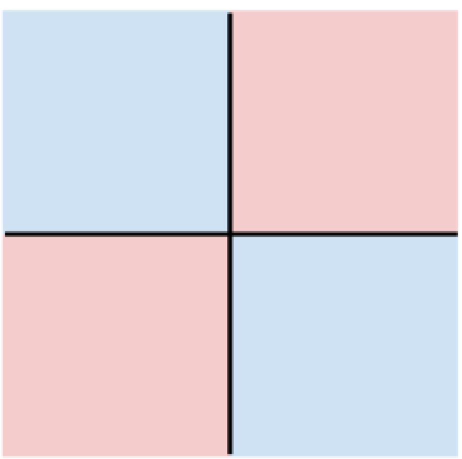
$$f(x,W)=Wx+b$$

32×32×3的数组
（共3072个数）

线性分类器的困难情况

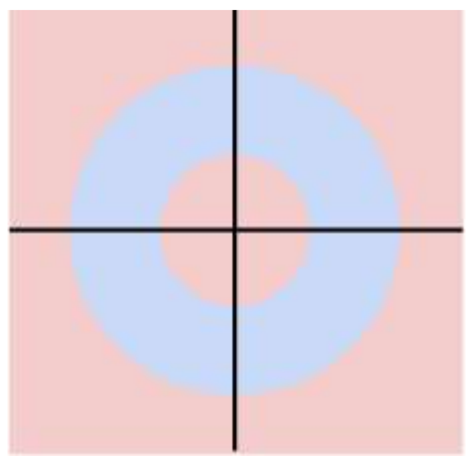
第一类:
第一象限和第三象限

第二类:
第二和第四象限



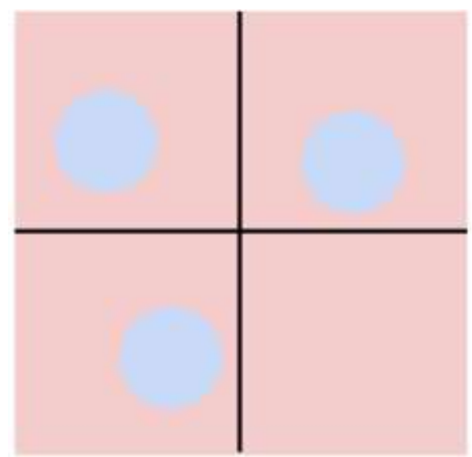
第一类:
1<=L2 范数<=2

第二类:
其余所有



第一类:
三种模式

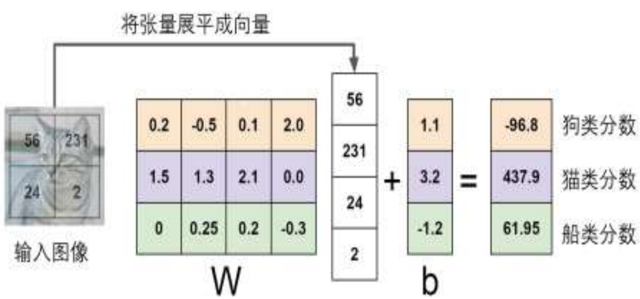
第二类:
其余所有



线性分类器：三种视角

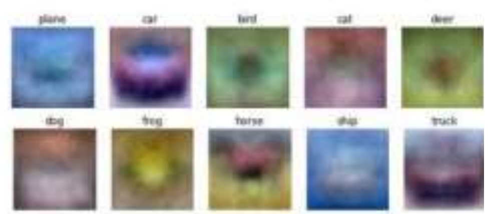
代数视角

f(x,W)=Wx



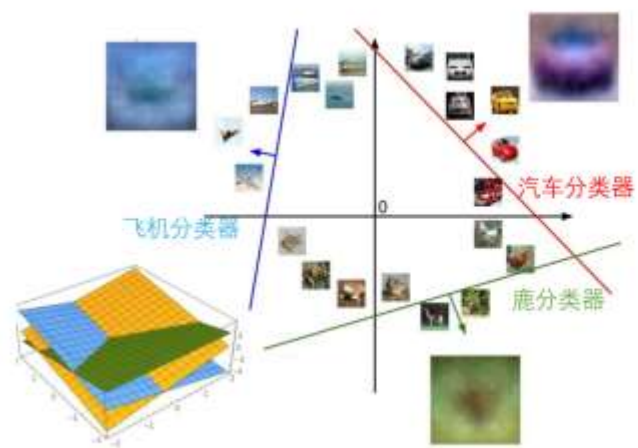
视觉视角

每个类一个模板



几何视角

超平面分割空间



$$f(x,W)=Wx+b$$

下讲重点：

-损失函数

（量化什么是“好”的W）

-优化器

（从随机的W值开始，找到一个使损失最小化的W）

-卷积神经网络

（调整f的函数形式）