Deep Learning

# 图像分类

其它一切视觉问题的基础

存在的挑战

* Viewpoint variation
* Scale variation
* Deformation
* Occlusion
* Illumination conditions
* Background clutter
* Intra-class variation

数据驱动

* 学习+分类
* 区别于固定的算法

The image classification pipeline

* Input
* Learning
* Evaluations

# 最近邻分类

L1\L2 distance

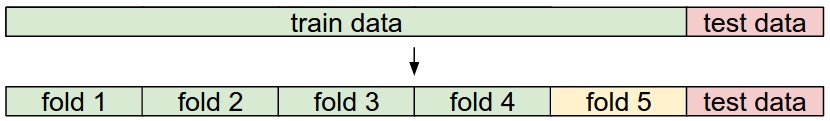
* L2 distance is much more unforgiving than L1 distance when it comes to differences between two vectors.

k-Nearest Neighbor Classifier

* 实际应用中KNN总是比NN要好
* K如何去取决于具体的问题

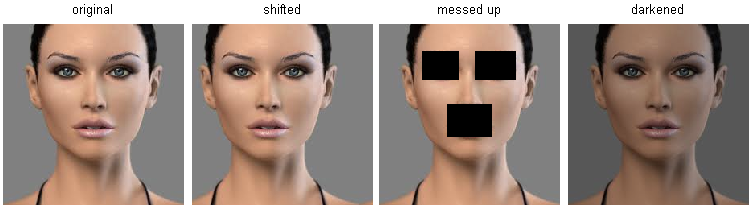
# 超参数设置

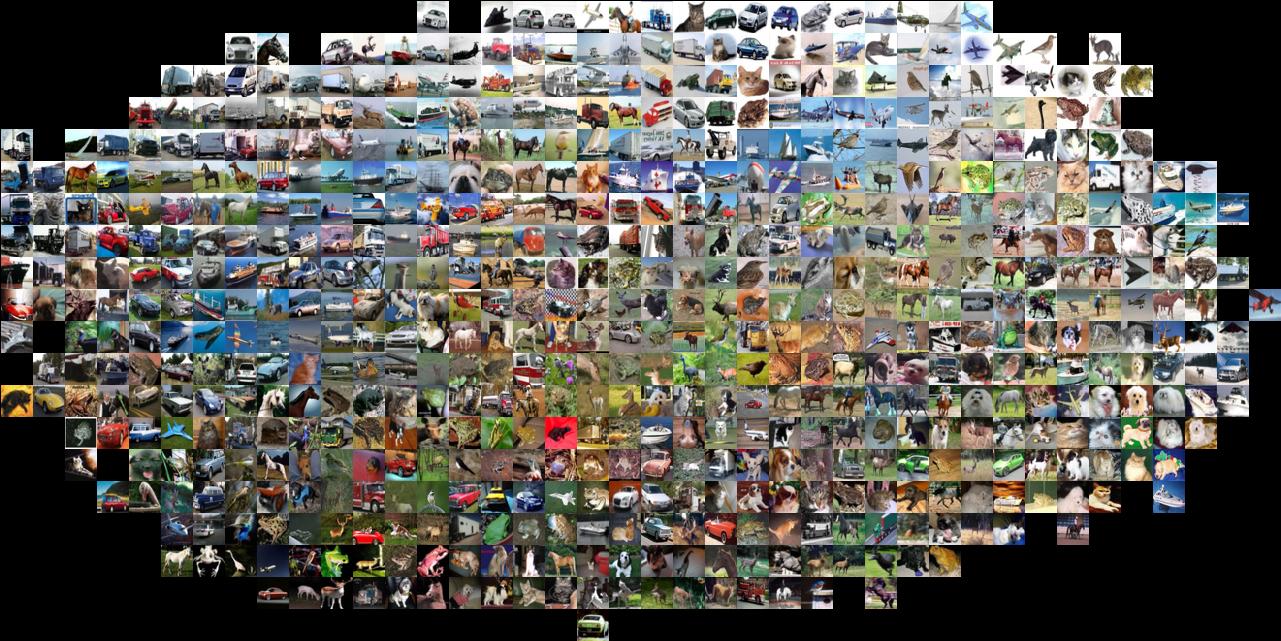
* Hyperparameter
  + K
  + Distance
* 不要用测试数据来修正超参数
* Cross validation（效果更准确，但计算量大）



# NN的优缺点

* Approximate Nearest Neighbor加速
* Images are high-dimensional objects, and distance over high-dimensional spaces can be very counter-intuitive.
* 距离的测量受到背景的影响更明显（通过t-SNE）例子。





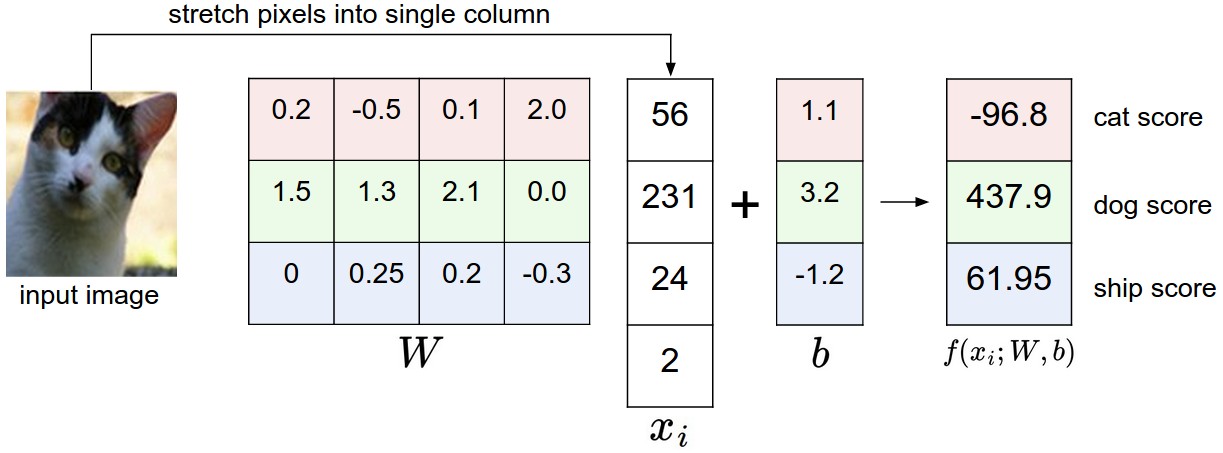
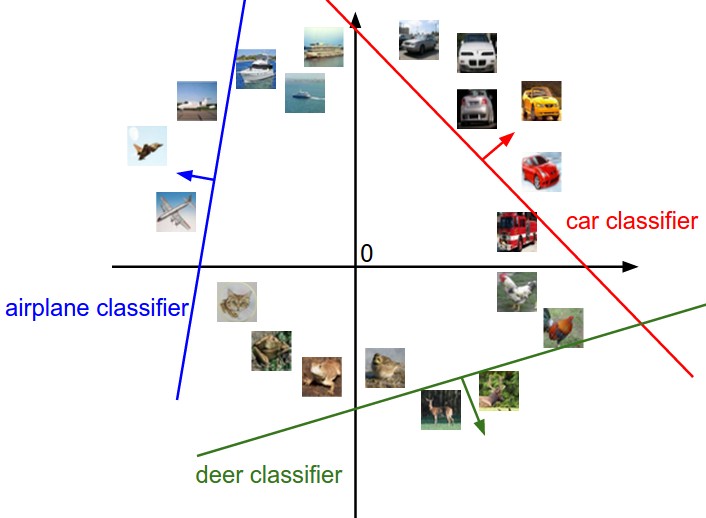
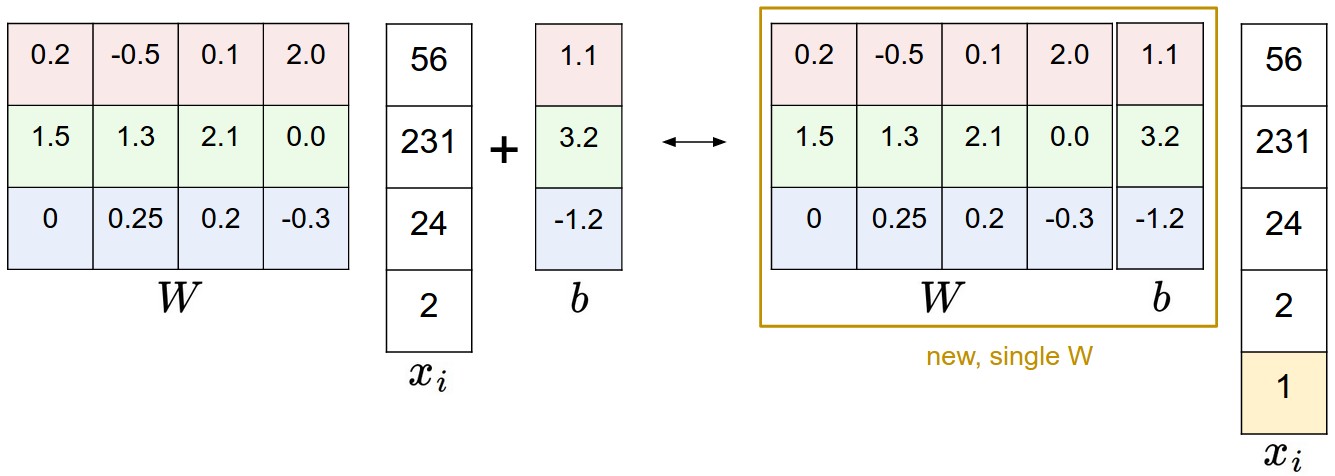
# 线性分类器介绍

* kNN的问题
  + 空间
  + 时间
* 新的方法核心问题：
  + Score function
  + Loss function

# Linear score function

* Linear classifier: f(x\_i,W,b)=Wx\_i+b
  + W: weights
  + b: bias vector
* Notes:
  + W描述n个平行的分类器，每个由其一行来表示
  + 数据是不变的，参数是不断改变的
  + 训练之后数据不再需要
  + 验证的过程只是一个矩阵相乘，比一个个比较要简单很多。

# 线性分类器的表示

* 矩阵视角：
* 
* 高维点视角：
* 
  + W控制分割面的角度，b控制分割面的位置（否则必经过原点）
* 模板视角
  + 类比于最近邻，但是每个类只有一个代表实例。
  + A neural network will be able to develop intermediate neurons in its hidden layers that could detect specific car types, and Neurons on the next layer could combine these into a more accurate car score through a weighted sum of the individual car detectors.
* 
* Bias trick
* 
* 数据预处理
  + Center your data
  + Normalization

# 损失函数

损失函数：

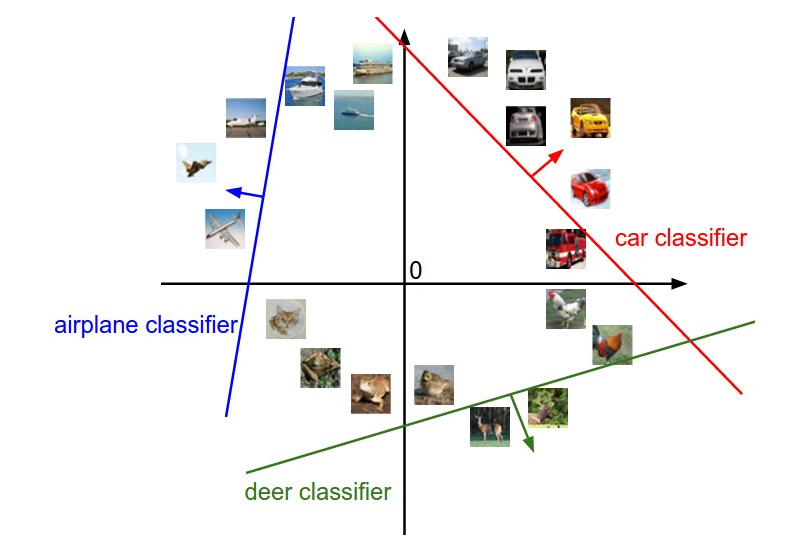
损失函数：

Binary SVM：

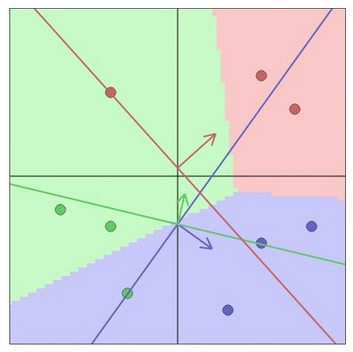
* , ()
* And
* So, , ()
* , ()

# 分类问题

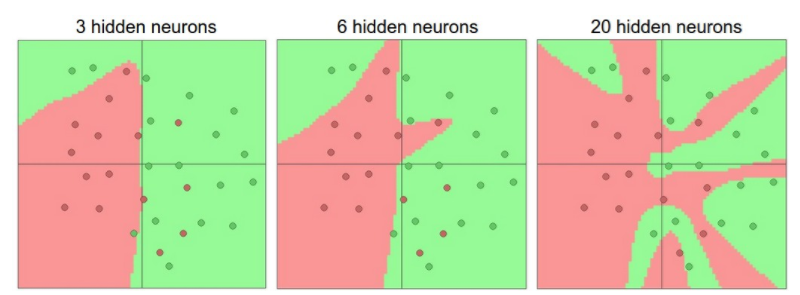
线性分类器：



神经网络：

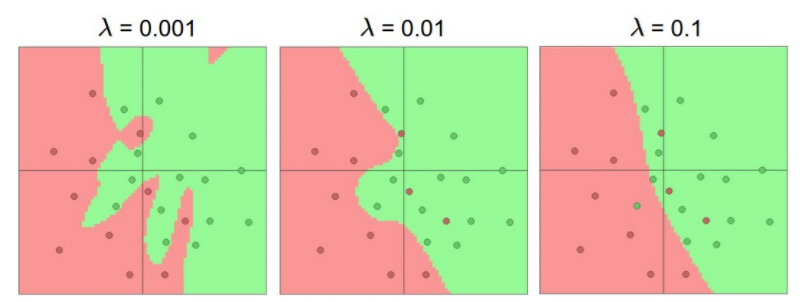


# 隐藏层的结点数量



在线例子：<http://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html>

使用Lambda控制过拟合问题。



# SoftMax

Loss function

]