cs231n

xzx

2016 年 12 月 2 日

Q1: k-Nearest Neighbor classifier

1 cs231n/classifier/k_nearest_neighbor.py

1.1 compute_distances_two_loops(self,X)

直接想比较困难, 题目给出提示说将其转化为矩阵相乘以及两个和的形式。于是想到将平方和展开。为方便起见, 先用平方距离进行计算, 计算完成以后再将其开方。设两个向量为 W 与 V。那么这两个向量的平方距离应该为 $(w_1-v_1)^2+...+(w_n-v_n)^2$ 将起展开,得到 $x_1^2-2x_1y_1+y_1^2+...+x_n^2-2x_ny_n+y_n^2$, 继续变形，得:

$$\sum_{i=1}^{n} x_i^2 + \sum_{i=1}^{n} y_i^2 - 2\sum_{i=1}^{n} x_i y_i$$

这个式子的前两项就是题目中说的两个和的形式，后一项就是矩阵相乘的形式.

Q2: Training a Support Vector Machine

2 cs231n/classifier/linear_svm.py

2.1 svm_loss_vectorized(W, X, y, reg)

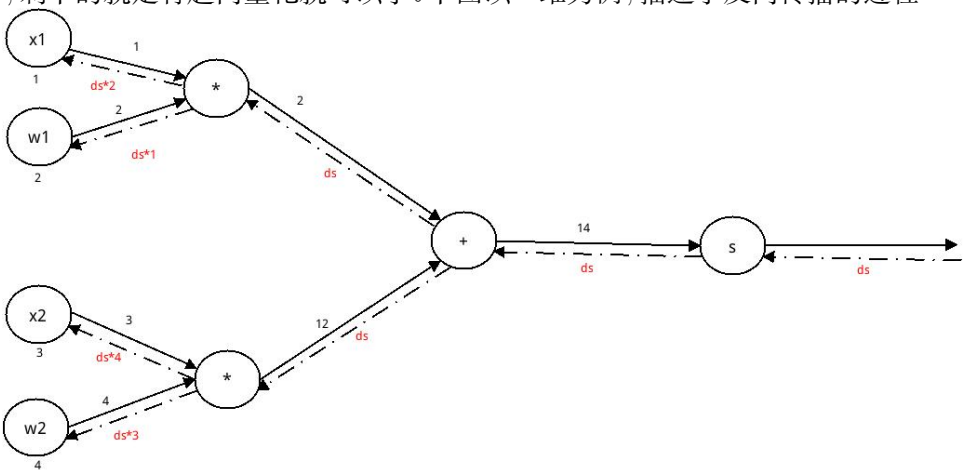
2.1.1 loss

首先把 loss function 的公式给列出来: $L_i = \sum_{j\neq y_i} max(0, s_j - s_{y_i} + 1)$ 不管怎么样, 肯定是要知道每一个图片对应的每一类的概率. 先用 $X * W$ 把

分数算出来。然后取出正确的分类的分数，相减.python 中对于矩阵减去向量，如果维数正确的话，它会自动补齐的. 然后再加上 δ，最后把负数全部都变成 0 就好了.

2.1.2 微分

首先不考虑要除以的 num_train。首先看公式中 s_j 与 s_{y_i} 对 loss 的影响。当 $s_j - s_{y_i} + 1$ 小于 0 的时候，对 loss 是没有影响的。当大于 0 的时候，如果 s_j 变化 1，则 loss 相应的变化 1；如果 s_{y_i} 变化 1，loss 相应的变化-1. 这样，ds 就可以表示出来了. 再来考虑每一个 s_i, j 是怎么来的。假设说 X 是 (N, D) 的矩阵，N 表示数据的个数，D 表示每个数据的维度, W 是 (D, C) 的矩阵，C 表示类型的个数. $s_{i,j} = X_{[i,:]} * W_{[:,j]}$. 如果 $W_{[t,j]}$ 变化了 1，则 $s_{i,j}$ 相应要变化 $X_{[i,t]}$. 所以，$dW_{t,j}$ 就等于 $ds_{i,j} * X_{i,t}$, 剩下的就是将起向量化就可以了。下图以 2 维为例, 描述了反向传播的过程



当然 W 会对不同的 X 乘很多次。对于 W 不同的分支来说，就是不同位置导数相加就是一个 W 的导数.