本节将从介绍 SVM开始

从上节的‘线性分类器’：,作为引入神经网络的第一块基石

最的损失函数： ，为所有损失函数的平均值（对总样本数的）

：这里是给的图像 ：图像对应的label

多分类的SVM ~ 2分类的SVM

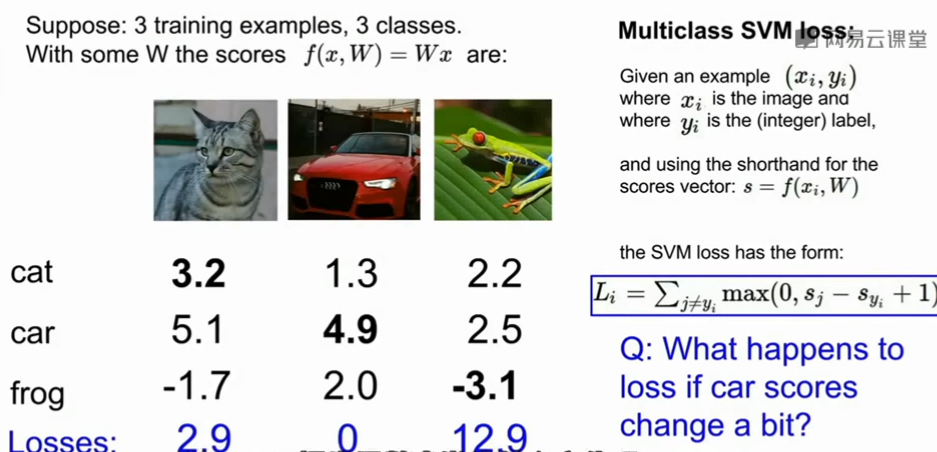


计算对每一个对应的预测错误结果：人总‘分数

问题1 ：汽车图片的得分分别为 1.3（猫） 4.9（汽车） 2.0（青蛙），

如果对汽车的分数做小的修改，结果会怎么样？

--结果不变

’

问题2：SVM损失函数的值域是什么？

--值域 为

问题3：当对于初步给定的W, 如果在实验初期，W都给的很小，并且很平均，如果所有输入x的score值，都近乎为0，那么Li的结果是什么？

--结果是n-1 , 这里的n是所有label值的数量

1. 求score值 f(x,W)



1. F求Li



1. 求L

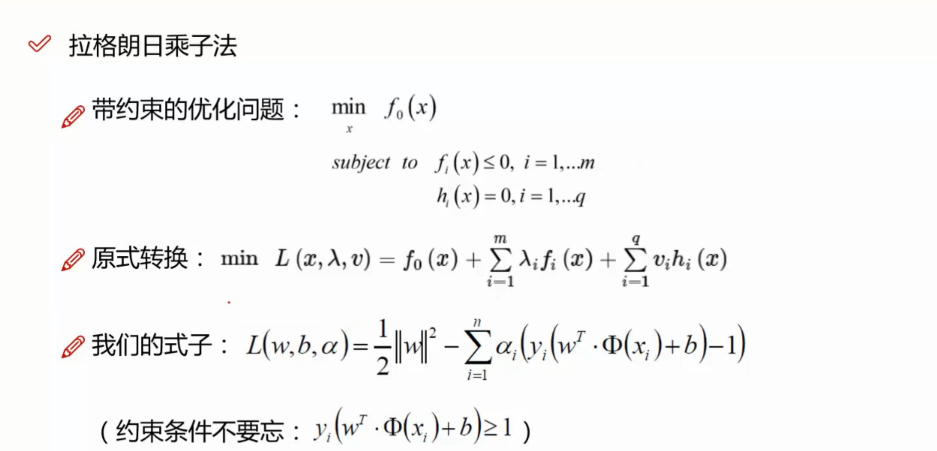


梯度下降时，一般用解析计算（analytic gradient），验证时用代数计算（umerical gradient）

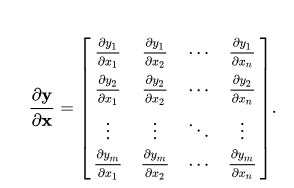
Always use analytic gradient,

but check implementation with numerical gradient.

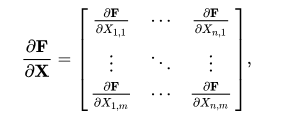
This is called a gradient check.



向量求导：

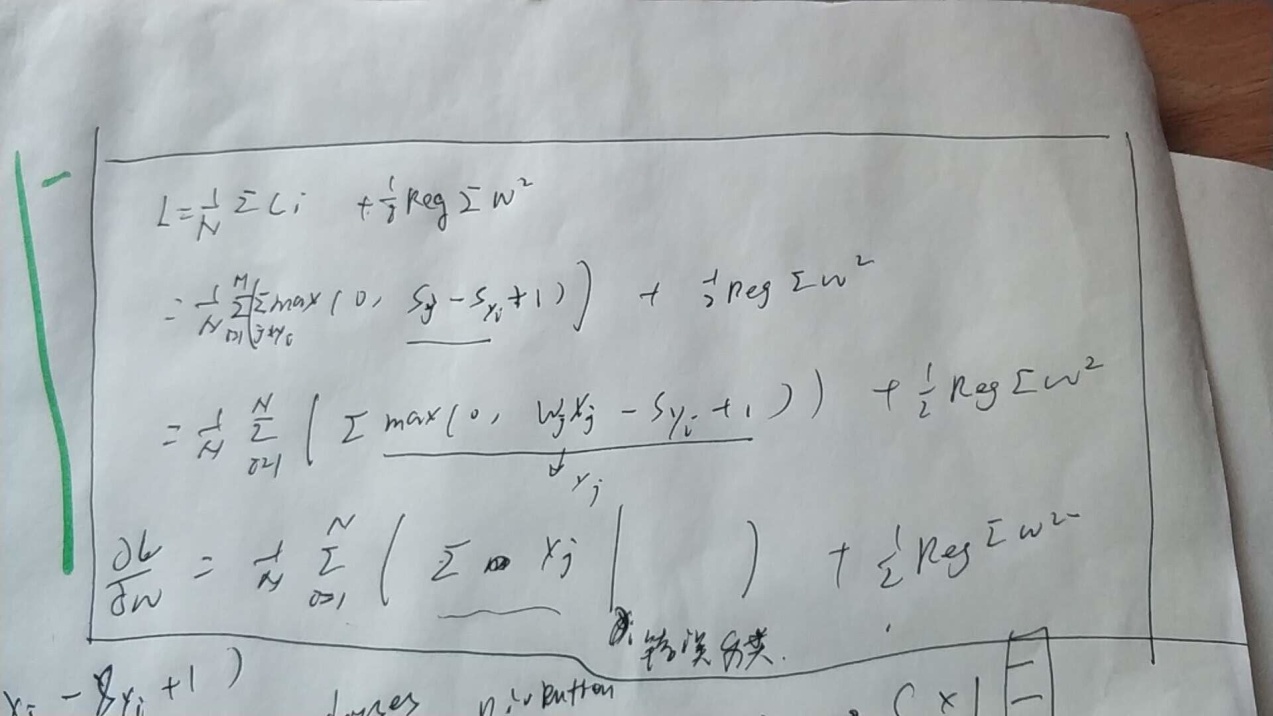


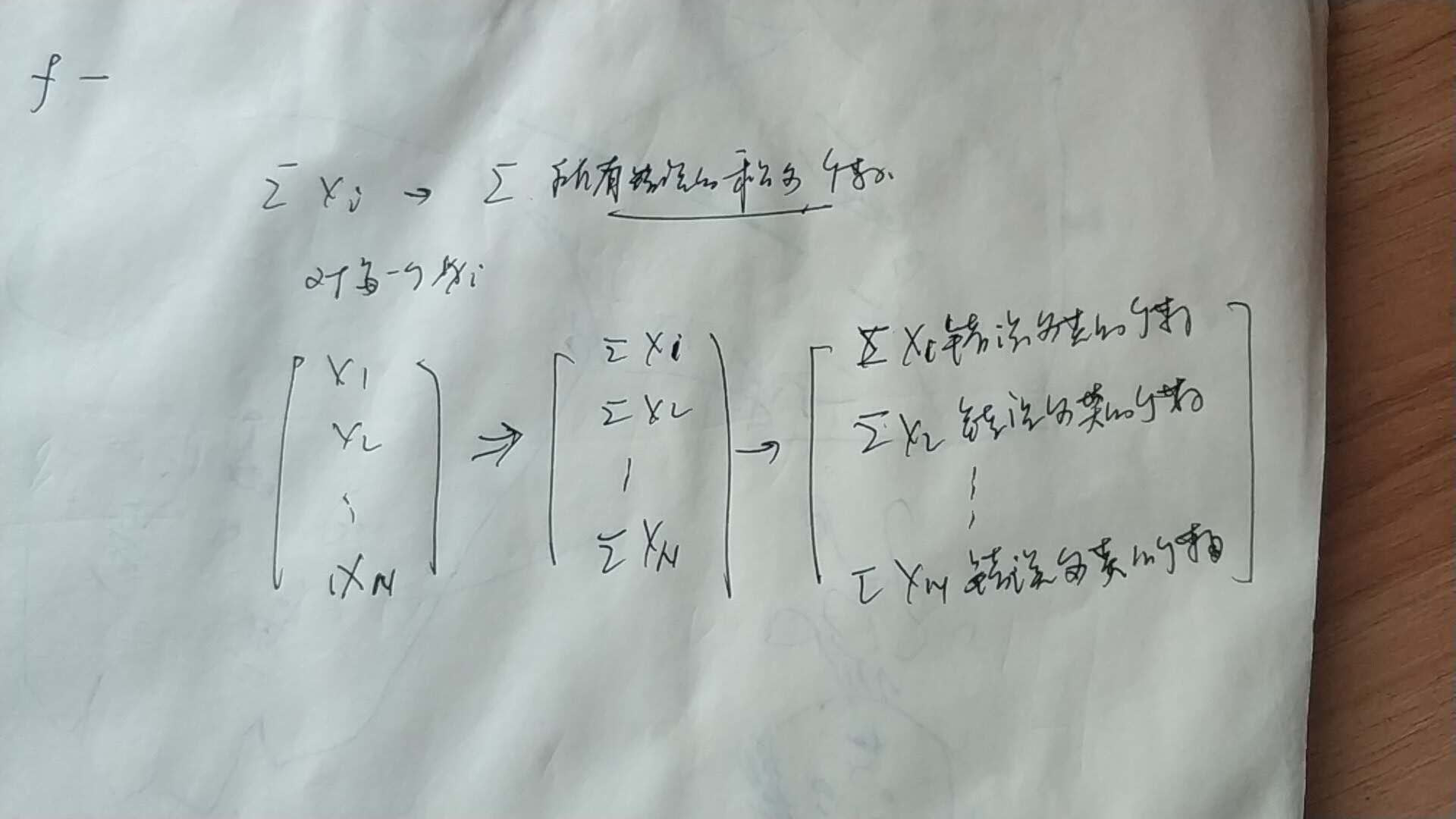
**矩阵求导**

****

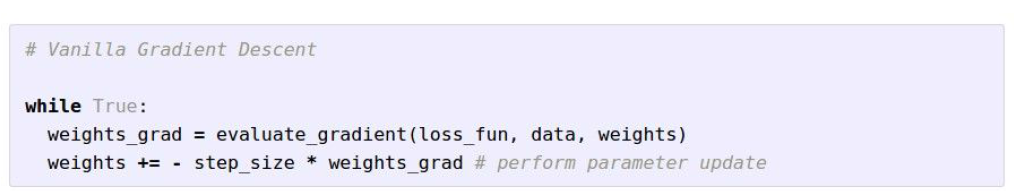
https://en.wikipedia.org/wiki/Matrix\_calculus

**对于每节的线性SVM,梯度推导：**



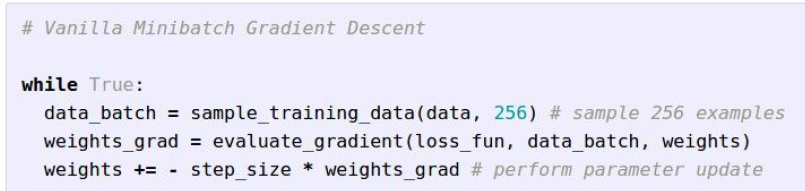
****

**一般梯度下降**

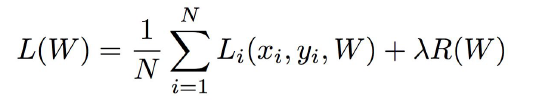


**随机梯度下降**

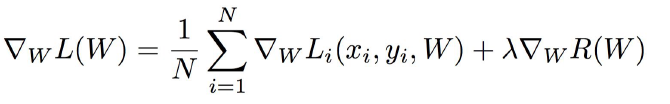
Stochastic Gradient Descent (SGD)



损失函数：

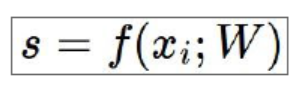


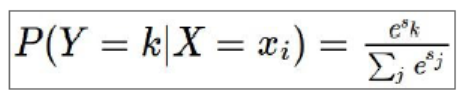
梯度：



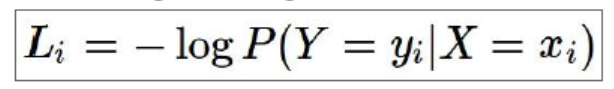
Softmax

S=f(xi;W) => P => Li

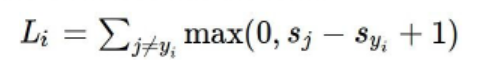


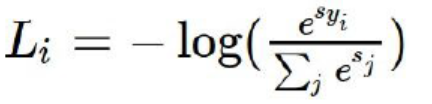


对输入的每一个x,正确标签对应的score 占所有score比例

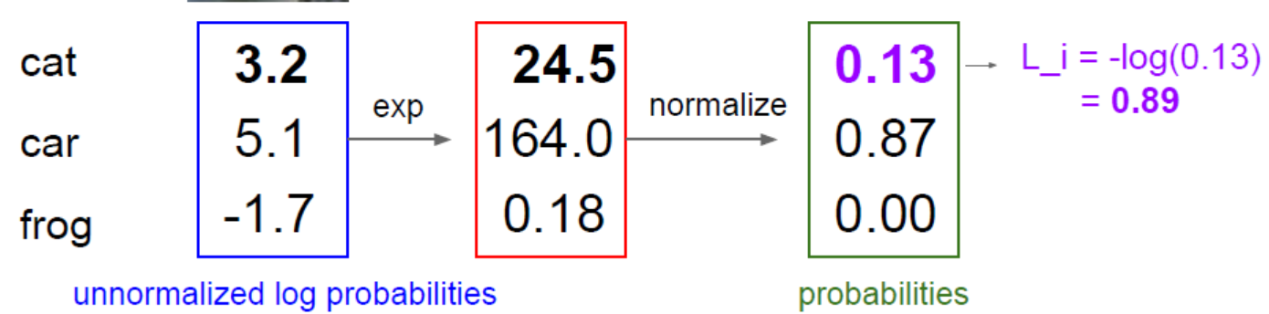


Softmax 与 SVM 的损失函数

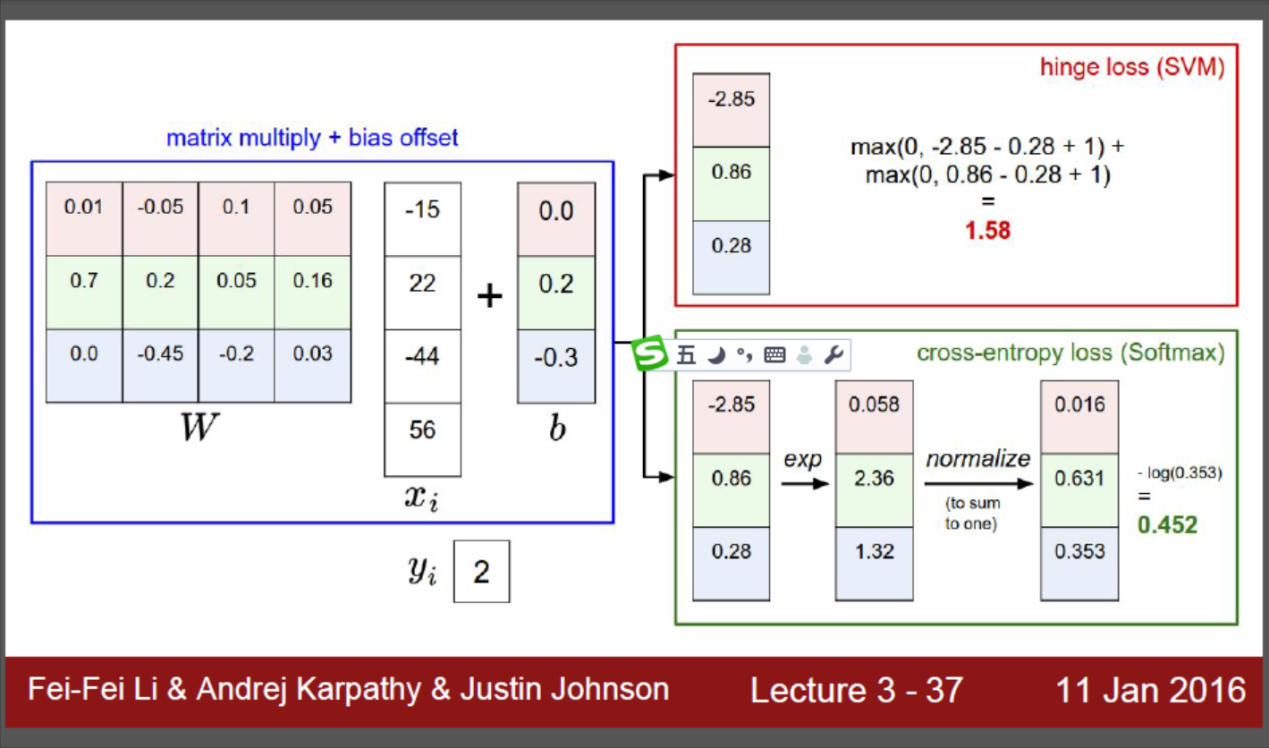
SVM: 

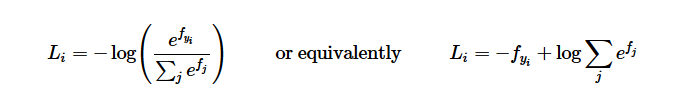
Softmax: 

Softmax的 计算公式 ：score->Li->normalize(归一化)



SoftMax 和 SVM 求LOSS的不同

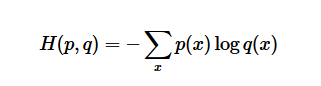




信息论的观点：

交叉熵

**Information theory view**. The cross-entropy between a “true” distribution *p* and an estimated distribution *q* is defined as: ‘真正分布’p，‘估计分布’q



Softmax 分类器是最小化交叉熵

其中，

P代表正确分类的概率 (i.e. *p*=[0,…1,…,0]) ， 在正确的分类为1，其他全为0

交叉熵又可以写成熵和KL距离的形式， **

而H（p）=0,(p 是 delta函数，要么是0，要么是1 p\*log(p) = 0)

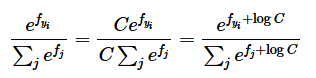
所以与最小化两个分布(p和q)KL距离等价

换句话说： the cross-entropy objective wants the predicted distribution to have all of its mass on the correct answer. 希望预测的结果都在正确的分类上

* Gaussian prior高斯先验

实际问题：**Numeric stability**. 数据稳定

实际中 可能很大，而除大数有可能得到的结果震荡过大，所以要归一化(normalization): 任意取一个常数C,下式永远成立

，我们可以通过C改进结果的不稳定性，通常的做法是取C, 使得 ，即最高得分的，然后让所有的score减去这个最大的score，这样使得新的score矩阵中最大的值是0，最小的值就是负振幅，代码如下：

