看到这一课，感觉这个系列文的日更计划又会延期，因为似乎该补一补CS229留下来的坑了，一知半解是最大的敌人。

以及，这个课程的第一个Assignment也在公开课中布置下来了，所以顺便重温一下Python和Numpy，去跑一跑实例。（coding才是入坑的标志）

闲话就先扯到这儿，开始第三课的笔记。

这一课的主要目标有：

1. **损失函数**（Loss Function, 下文简称LF）：量化我们对训练结果的满意程度，换句话说，是衡量分类器的错误程度。
2. **优化**（Optimization）：快速找到使得LF最小化的参数，以提高我们对分类器的满意度。

一、损失函数（Loss Function）

1. MULTI-CLASS SVM (SUPPORTED VECTOR MACHINE)

SVM损失函数具有如下形式：

$$
\begin{aligned}
L_i &= \sum_{j \neq y_i} \begin{cases} 0 & if \quad s_{y_i} \geqslant s_j + 1 \\ s_j - s_{y_i} + 1 & otherwise \end{cases} \\
&= \sum_{j \neq y_i} max(0, s_j - s_{y_i} + 1)
\end{aligned}
$$

其中，$max(0, Sj - Syi + 1)$的写法初看容易产生困惑，因此使用分类的格式描述（if-then），其中「1」是判定是否分类准确的边界值（margin）。这样，损失函数看上去像摊开的书本一样，因此把这种形式的损失函数称为**hinge loss**（hinge意为“合页”）。

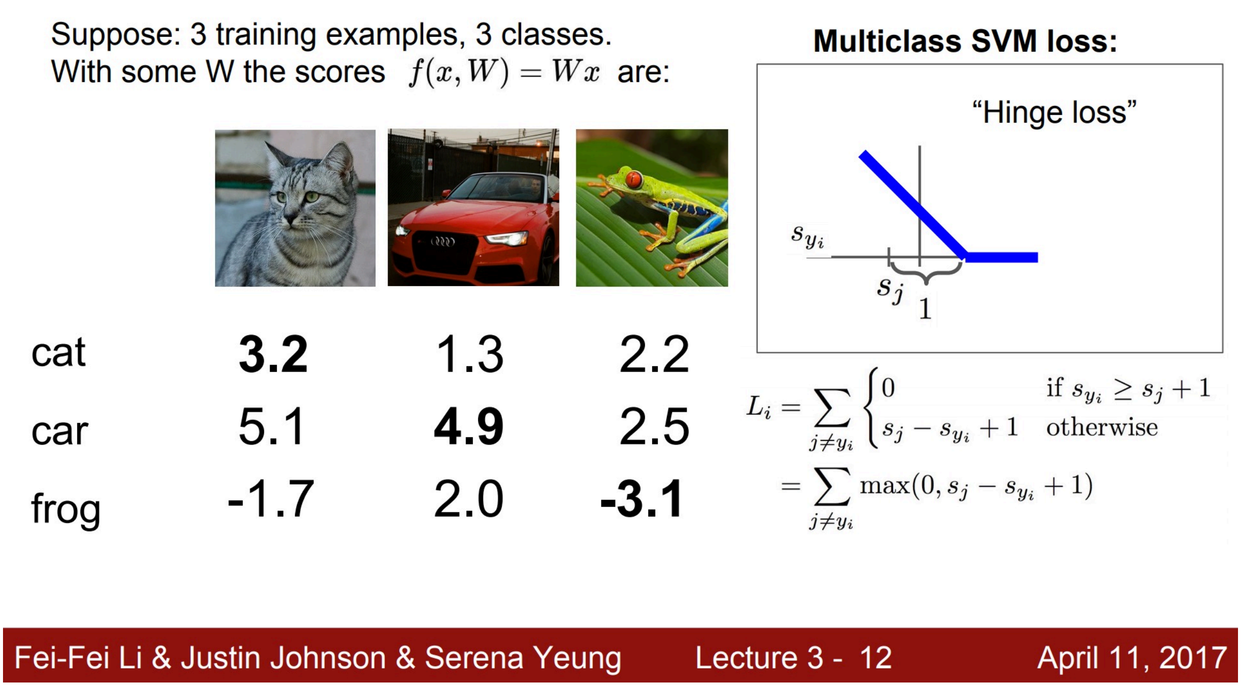
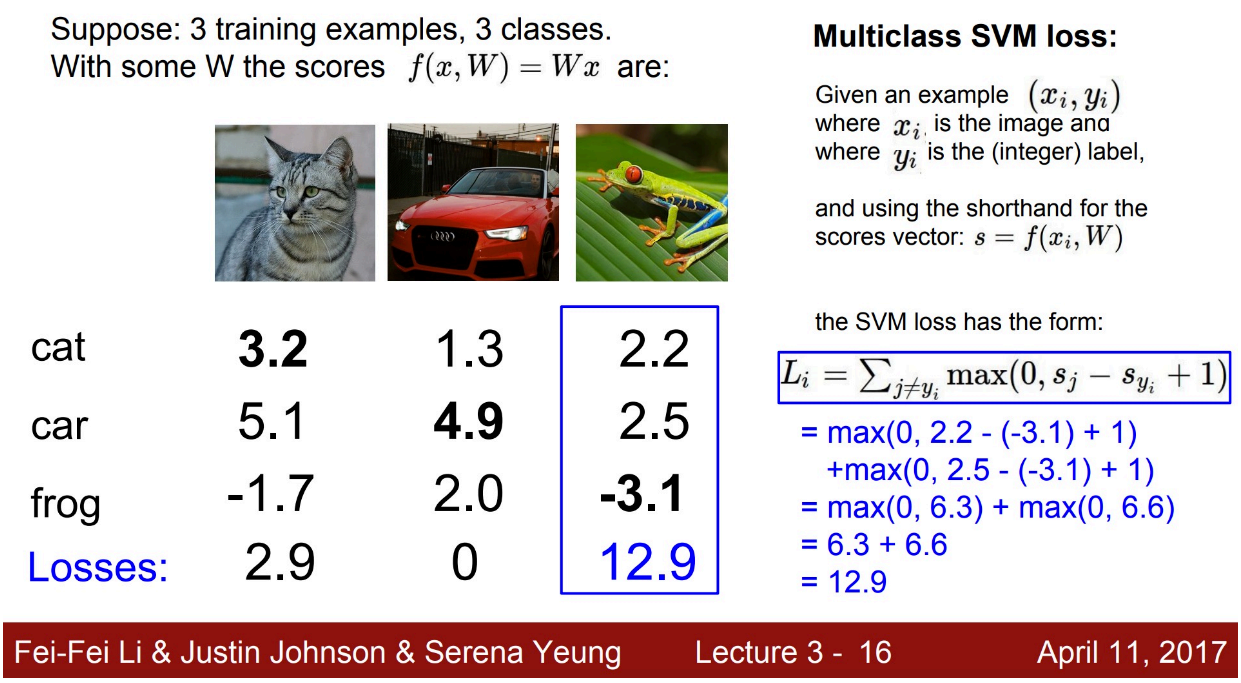


图1 Hinge loss

SVM损失函数的计算举例：



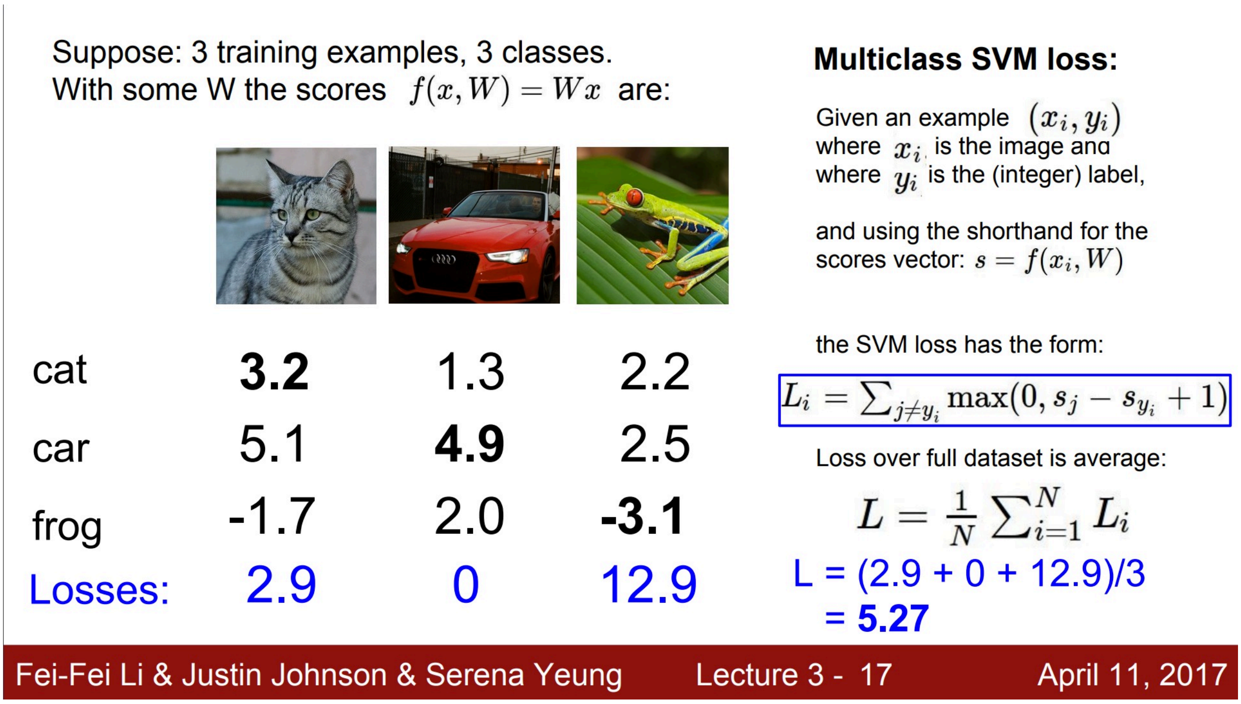


图2&3 SVM的计算

一些说明：

- SVM-LF在单类计算$W \cdot x_i$的结果值为0时，数据轻微波动不会对结果产生影响。
- 损失函数理论上计算得出最小值为0，最大值为无穷。
- 假设我们将W全部初始化为0，即结果$W \cdot x$为0，则SVM-LF计算结果为$C-1$（C为图像类别总数）。（可以利用这一点进行编码调试，即初次迭代结果应该为$C-1$。
- 如果使用平方量化损失，则会是完全不同的模型，平方量化更关心特别差的分类结果，而忽略一些不那么准确的分类结果。这一点同我们在前文中描述的模型在优化上的侧重点不同。

$$L_i = \sum_{j \neq y_i} max(0, s_j - s_{y_i} + 1)^2$$

- 使得SVM-LF的计算结果为0的W并不惟一，例如若W_1满足此条件，则$2W_1$也一定满足。

防止**过拟合**：对分类器进行调参时，并不一定要完全符合训练集，而是应该应用于未知的测试集。为此引入正规化函数，即对模型使用一些惩罚机制R(W)，使其不能完全匹配训练集。这满足奥卡姆剃刀的原则：“如无必要，勿增实体”。

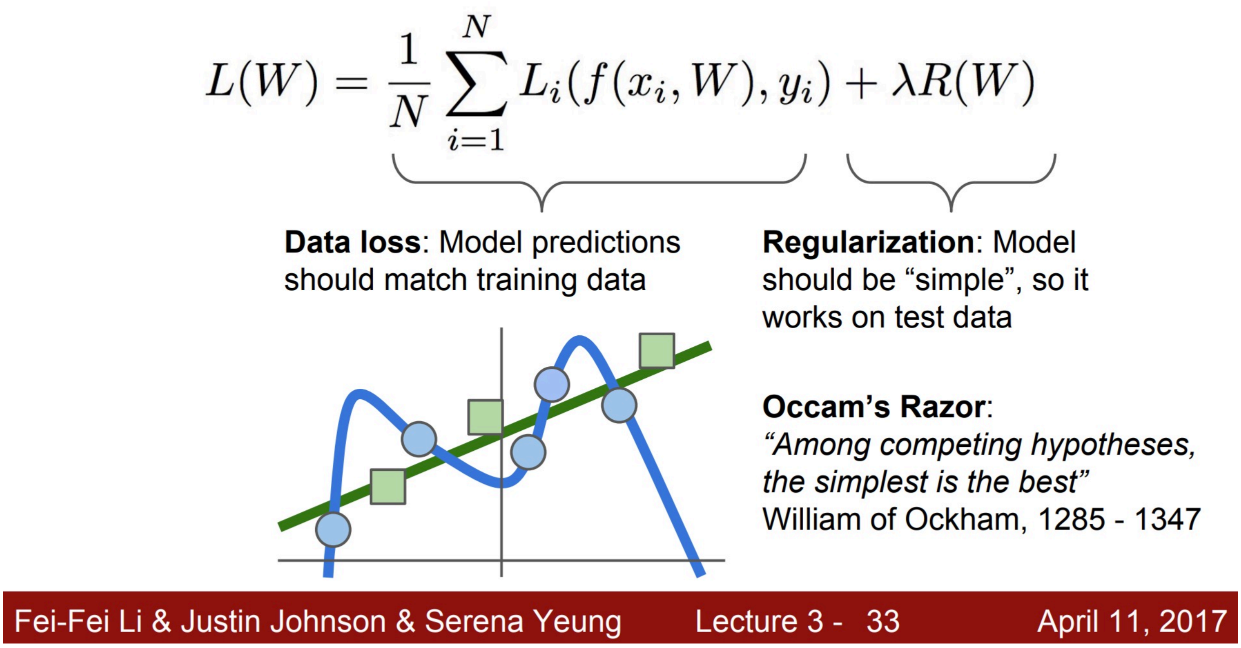


图4 过拟合及其惩罚

一些常见的函数R的形式如下：

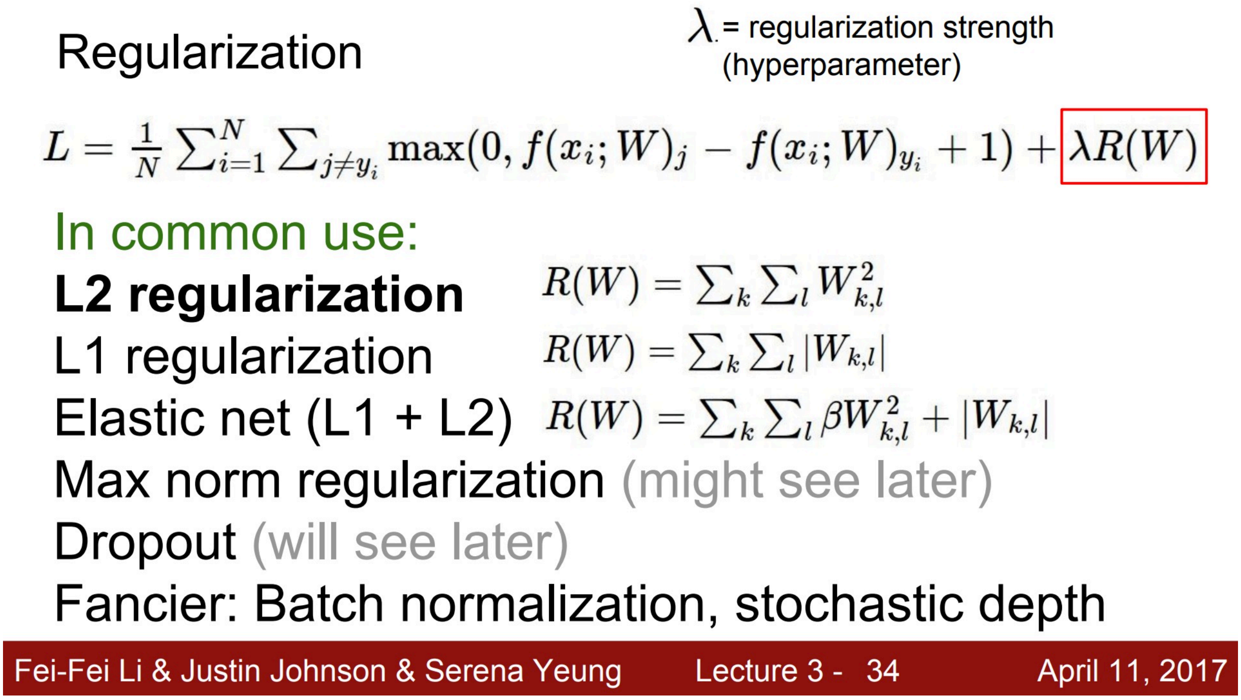


图5 惩罚函数R

2. SOFTMAX CLASSIFIER

参考Wiki上给出的关于Softmax的说明：

> In mathematics, the **softmax function**, or **normalized exponential function**,[1]:198 is a generalization of the logistic function that "squashes" a K-dimensional vector $\mathbf{z}$ of arbitrary real values to a K-dimensional vector $\sigma(\mathbf{z})$ of real values in the range [0, 1] that add up to 1. The function is given by:
>
> $$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$ for j = 1, …, K.

在了解Softmax函数的相关推导过程之前，可以先将其视为一种将分类器输出的Score映射成概率的函数。而基于Softmax的损失函数即为该概率的**负对数**，即：

$$L_i = -logP(Y = y_i|X = x_i)$$

Softmax计算示例如下：

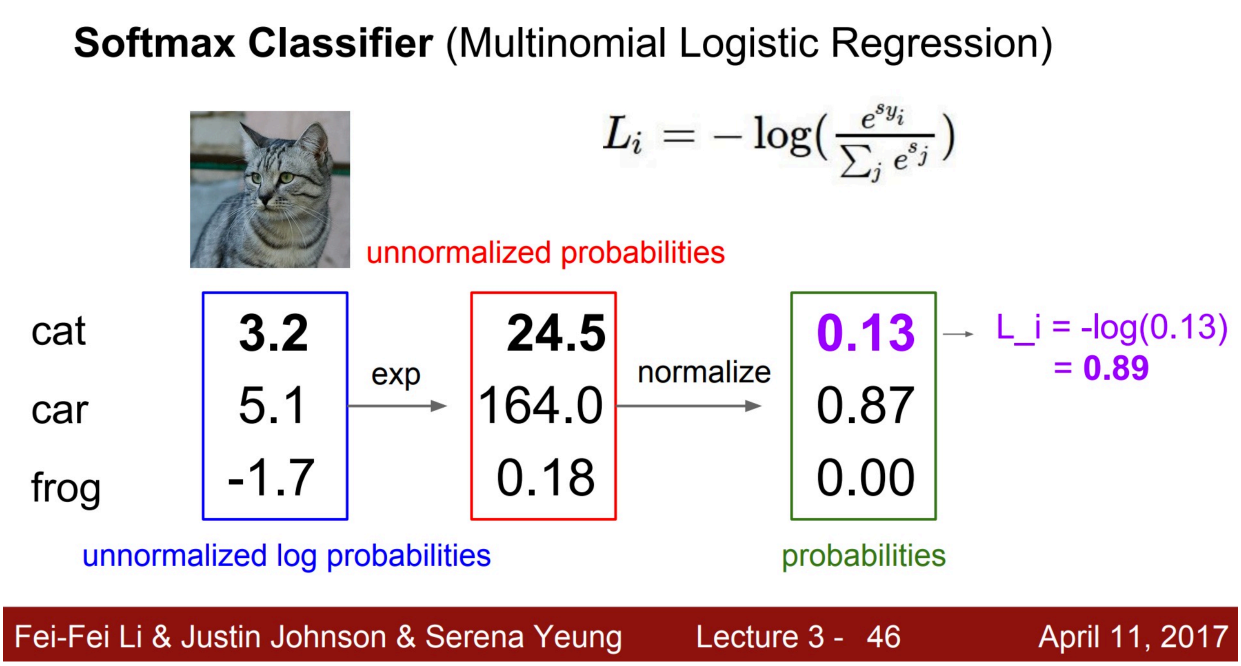


图6 Softmax计算示例

说明：若使用Softmax函数进行初次迭代，则所有score均为0，而$exp(0) = 1$，得到概率为$1/C$ (C为类别总数)，再取负对数，得初次迭代结果应为$-log(1/C) = logC$。

这样，介绍完了两种常见的损失函数，和用来控制过拟合的惩罚函数，得到以下的计算总损失的过程：

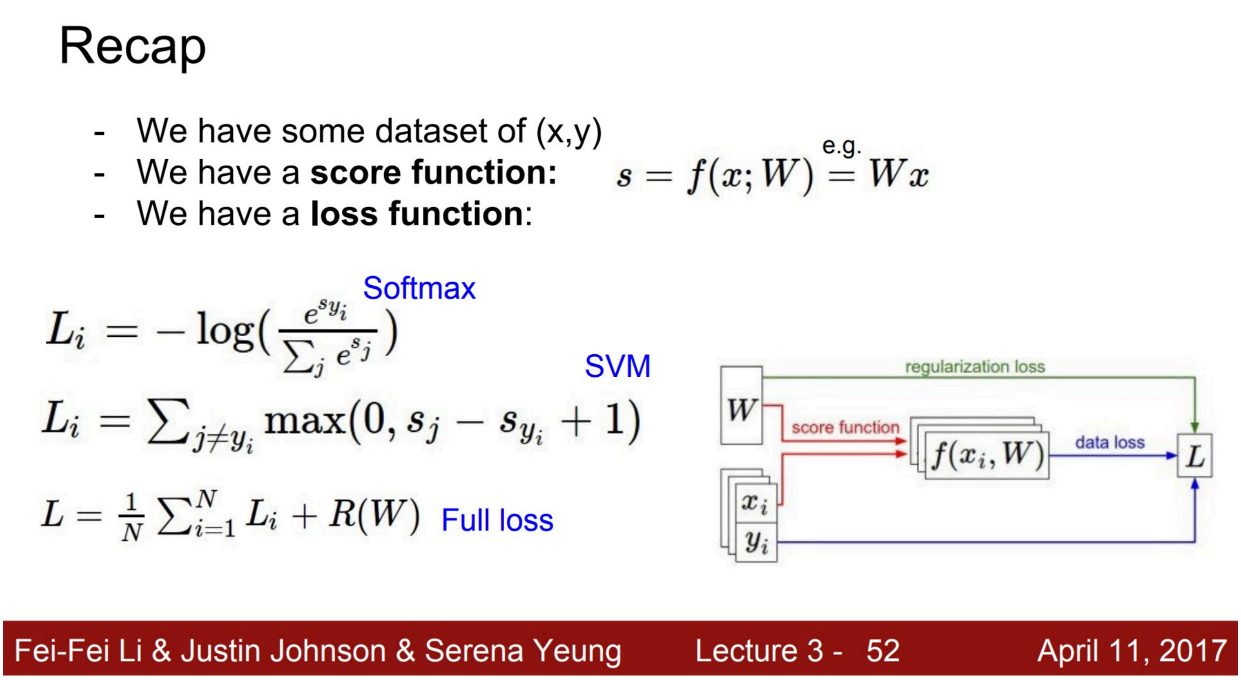


图7 总损失L的计算

那么，如何找到满足条件的W呢？这就是在第二节——优化(Optimization)中需要处理的问题。

二、优化（Optimization）

1. 随机搜索

这是一个很自然但是很不好的方法，不再赘述。

2.沿着坡走(Follow the Slope)

设想你处在一个山坡上，找到山谷最低点的方法自然就是用脚去感受四周的坡度，找到坡度下降最快的方向，然后迈上一小步，在新的位置，重复上述的方法再迈一小步，按照这种方法，（或许）就能找到山谷的最低点。

图8 沿着坡走

一般来说，坡度的计算分为单元函数和多元函数，前者可以直接求导，而后者则逐元求偏导。

求导可以按照定义得到**数值梯度**（Numerical Gradient），也可以用公式得到**分析梯度**（Analytic Gradient）。通常，定义式求导很慢且不精确，但可以用来**调试**，而用公式求导则较快且准确，但容易出错。

- 用数值梯度验算的过程称为**梯度检验**（Gradient Check）
- **步长**（ **step_size**，也称作learning rate），可能是线性分类器中最为重要的超变量，通常在训练模型开始前首先设置该参数。

下面介绍两个针对**计算速度**的优化算法：

（1）随机梯度下降法（SGD, Stochastic Gradient Descent）

观察之前得出的损失函数计算表达式：

$$\nabla_W L(W) = \frac{1}{N}\sum_{i=1}^{N} \nabla_W L_i(x_i, y_i, W) + \lambda \nabla_W R(W)$$

不难发现，当训练集内元素数量过多的时候（即N非常大），每进行一次迭代的代价都特别高。为此，引入**SGD**方法。即：从N中随机抽取32/64/128...个（2的整数次幂）样本进行损失函数L的计算，按照该计算结果进行梯度下降。

（2）提取图片特征（Image Features）

一张图片的大小通常在 几百*几百*3 的数量级，在这个数量级下进行矩阵运算仍然十分耗时。为此，我们可以提取图片的特征来替代原始的像素值信息。通常采用的特征有：

- 颜色直方图（Color Histogram）

这是一种统计方法，即把图片划分为多个颜色区间，统计属于不同颜色区间内的像素数量，得到一组特征值。

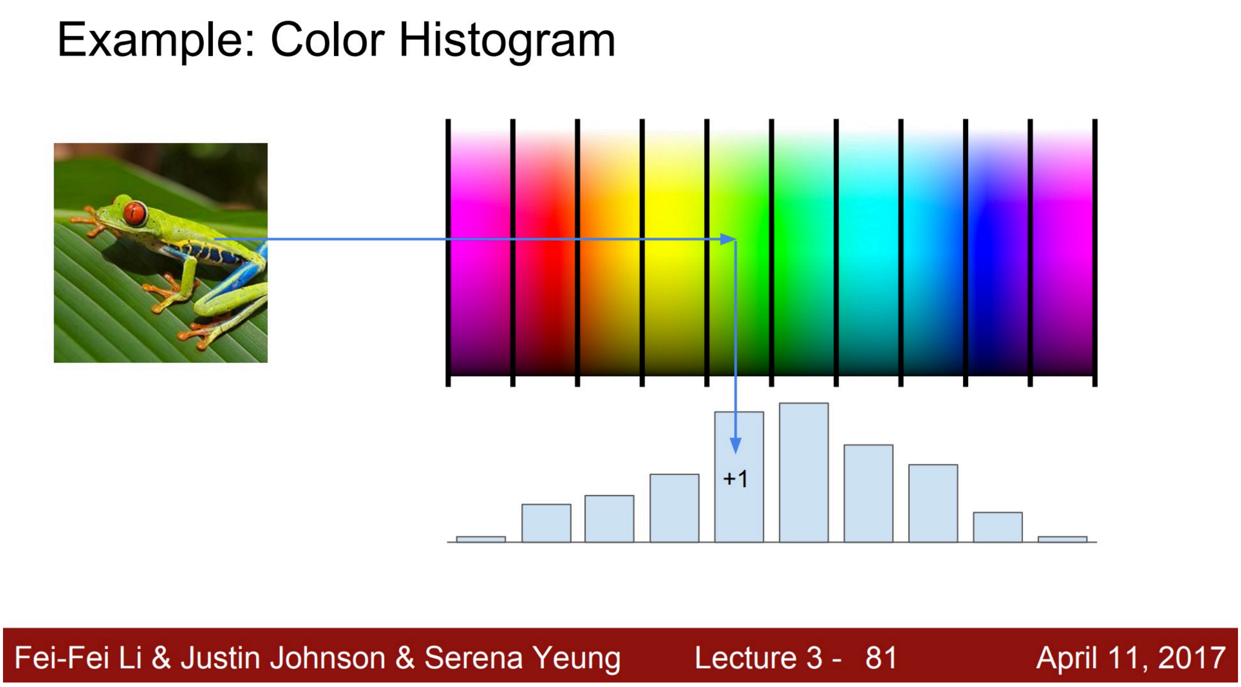


图9 颜色直方图

- 定向梯度直方图（Histogram of Oriented Gradient）

将图片分成若干小区域，根据区域内的颜色变化情况，描述该区域的梯度变化情况。

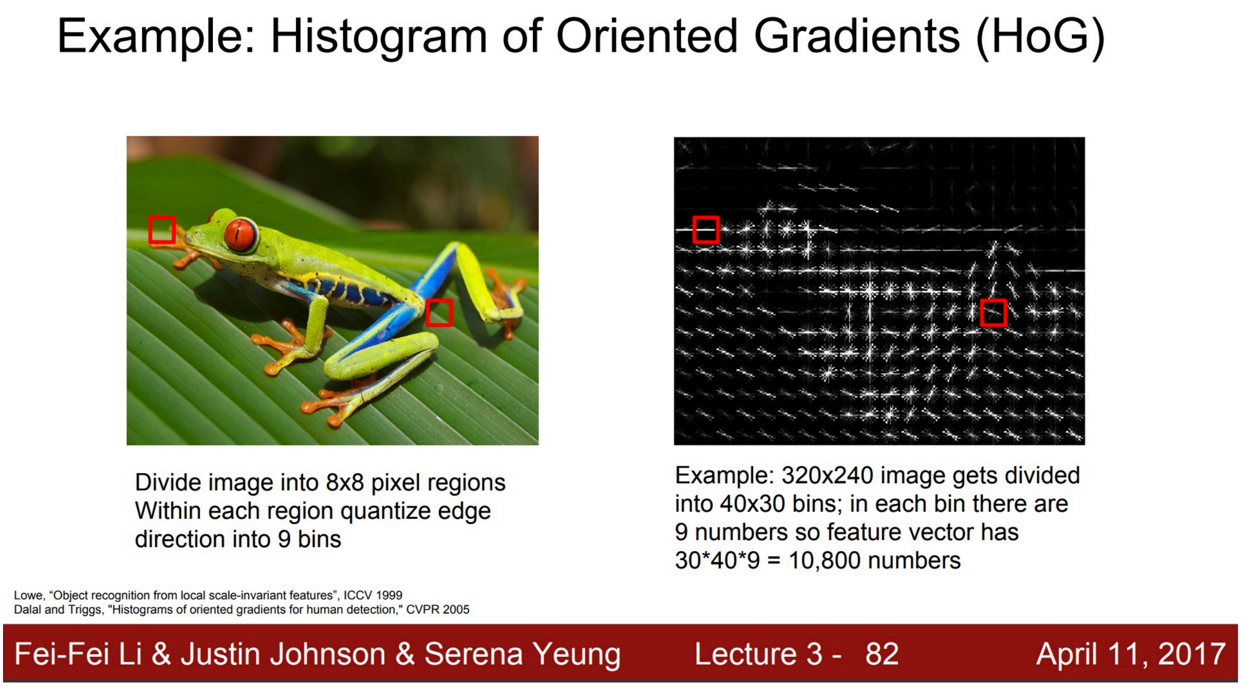


图10 定向梯度直方图

有关HoG的更多理解准备单独写篇文章整理。

- 词袋模型（Bag of Words）

词袋模型是来自自然语言处理领域的方法。为了表示一幅图像，我们可以将图像看作文档，即若干个"视觉词汇"的集合，同样的，视觉词汇相互之间没有顺序。将其应用到图像处理后，第一步同HoG类似，也是将图片分成若干小区域，然后对这些分离的小区域进行聚类（运用k-means算法等）。

建立了图像词袋后，我们对图像根据词袋编码，得出一组统计特征值，即为这个图像的另一种特征表达。

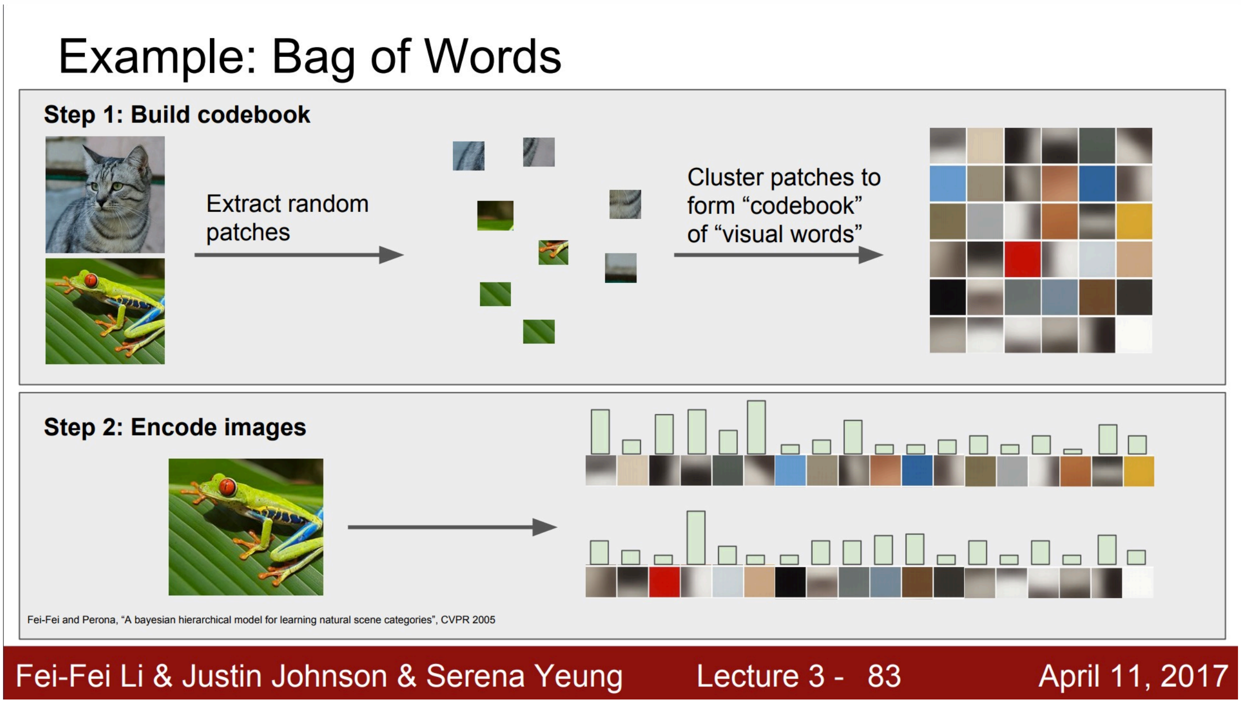


图11 词袋模型

有关BoW的模型也可以开一篇文章整理。

按照以上方法提取图片特征后，原本图片中数以万计的像素值矩阵，变成了较少的一些特征值，通过调整这三种特征的权值，在保证准确性的前提下，可以大幅度提升分类器的运算速度。

CS231n-3的笔记整理就到这里，尚存不少疑点需要进一步理解。同时也可以进行相关编码任务或者继续刷CS229了。

By SillyDog.