Lecture 11 Linear Models for Classification

整理者：LobbyBoy* 2020年2月25日

1. Linear Models for Binary Classification

无论是linear classification，还是linear regression，还是logistic regression，它们的hypothesis都需要计算一个score：

$$\mathbf{w}^T\mathbf{x}$$

然后让score经过一个函数，得到最终的输出。不同的是，linear classification为sign函数，linear regression为线性函数，logistic regression为sigmoid函数。此外，训练它们时选择的错误度量也各不相同。具体对照如下图所示：

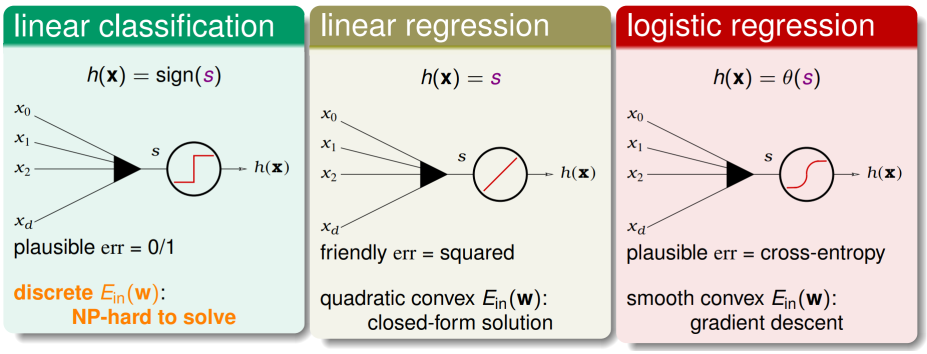

图 1: Linear models

我们知道，直接最小化0/1误差的问题是NP-Hard问题，无法求解。因此我们此前用解线性回归的方式帮助解决二元分类问题，并推导出了理论保证，即平方误差是0/1误差的上界，平方误差做得很小也就说明0/1误差也被做得很小。下面，我们将把0/1误差与平方误差和交叉熵误差放在一起，探究它们之间的关系，并得到logistic回归可以用于解决二元分类问题的理论保证。

令$\mathbf{w}^T\mathbf{x} = s$，对于三种模型，我们都可以将单个样本的误差写成$ys$的函数：

*本笔记根据台湾大学林轩田教授于线上教育平台Coursera开设的“机器学习基石”课程整理而成(课程内容见：https://www.coursera.org/learn/ntumlone-mathematicalfoundations/home/welcome)。笔记内的大多数图片来自于林老师的课程slides。感谢林老师能够将如此精彩的课程通过线上平台同所有人分享，thanks!

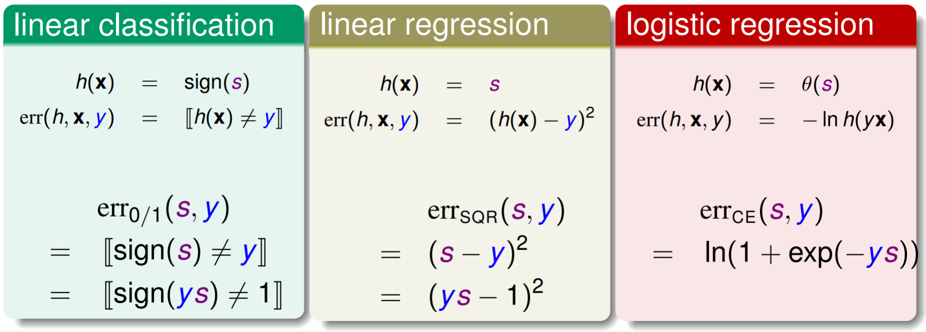

图 2: ys

ys被称为classification correctness score，因为ys越大→越正→同号→预测方向相同→预测正确。下面我们以ys为横轴，error为纵轴作图，将三种误差画在一个坐标系内：

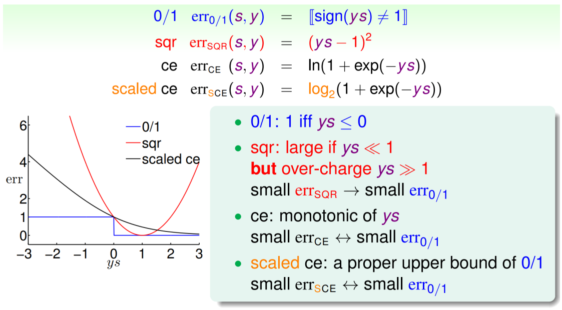

图 3: Upper bound

和前几章的结果一样，平方误差是0/1误差的上界，且平方误差在$ys = 1$附近与0/1误差十分接近，在两端则与0/1误差相差甚远。特别在$ys >> 1$时，0/1误差下该样本点并未犯错，但在平方误差下该样本点却被认为犯了很大的错误。对于交叉熵误差，我们将其除以$ln(2)$进行换底，得到了“scaled交叉熵误差”。这样进行放缩的原因是：放缩后的交叉熵误差正好过点$(0, 1)$，成为0/1误差的上界。因此我们有：

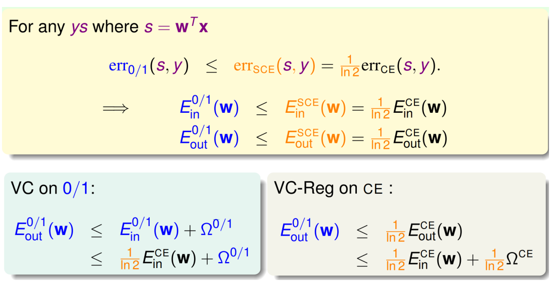

图 4: Guarantee

因此，与线性回归→线性分类类似，如果我们把交叉熵误差做得很小，那么该hypothesis对应的0/1误差也很小，我们就可以直接将其$\mathbf{w}$拿来作为线性分类器使用，即：

经验上，我们常常用linear regression跑的结果作为PLA/pocket/logistic regression的初

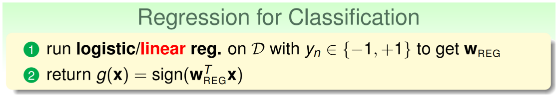

图 5: Regression for classification

始值；且常常更多使用logistics regression而非pocket。

2. Stochastic Gradient Descent

梯度下降法的核心是不断地进行$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta \cdot \mathbf{v}$，其中$\eta$为fixed learning rate，$\mathbf{v}$为负梯度。对于logistic regression，迭代式如下：

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta \cdot \frac{1}{N} \sum_{n=1}^{N} \left[\theta(-y_n \mathbf{w}_t^T \mathbf{x_n}) \cdot (-y_n \mathbf{x_n}) \right]$$

上式中的累加求和部分是经验误差的梯度，我们看到出现了$\frac{1}{N}\sum_{n=1}^{N}$。因此，我们有理由将该梯度看作是“某些项”的uniform expectation，这里的“某些项”为：

$$\theta(-y_n \mathbf{w}_t^T \mathbf{x_n}) \cdot (-y_n \mathbf{x_n})$$

我们将其称为stochastic gradient，记作$\nabla_{\mathbf{w}} err(\mathbf{w}, \mathbf{x}_n, y_n)$，其中$n$是随机的，可以取1到$N$的任意值，且取每个值的概率相等(uniform)。易知，该stochastic gradient的期望为原梯度，我们也称之为true gradient。我们看到，stochastic gradient只与某一个样本有关，而true gradient则要平均每个样本的贡献，因此我们自然想到在梯度下降时用stochastic gradient代替true gradient。这样做尽管可能在一轮迭代中产生一些偏差，但是如果迭代次数很多的话，正负误差会几乎抵消掉，所以终点与普通的梯度下降并不会相差很远。更重要的是，用stochastic gradient下降能够节约大量的算力，使学习过程更加快速；同时，这也使得online learning成为可能——对于新添加的一个训练样本，我们可以用stochastic gradient来更新当下的hypothesis。综上，我们将这种方法称为stochastic gradient descent。SGD logistic regression的迭代式如下：

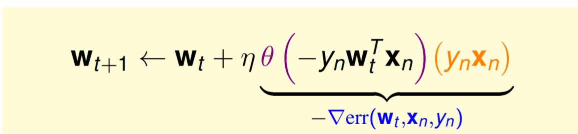

图 6: SGD logistic regression

使用SGD logistic regression的两条经验法则：①步长取0.1；②t足够大时停止迭代输出结果，而不是在每一轮计算梯度看梯度是否够小。

3. Multiclass via Logistic Regression

此前我们解决的分类问题都是二元分类问题，即回答“Yes or No”的问题。我们能否在此基础上，解决多元分类问题？如下：

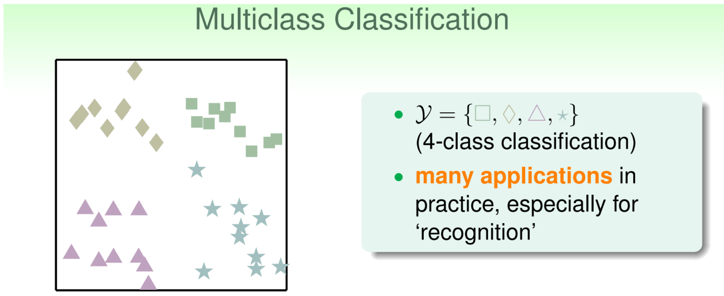

图 7: Multiclass classification

一种容易想到的方法是：One Class at a Time，即为每一个类别训练一个二元分类器(在训练某类别的分类器时，将该类别的样本标注为+1，其余类别的样本标注为-1，然后应用我们学过的二元分类方法)，然后将它们结合起来，用以判断某个样本的输出类型。这个思路乍一想没有问题，但仔细思考会发现，将各个类别的分类器结合起来的时候，会出现“真空地带”或“冲突地带”，如下：

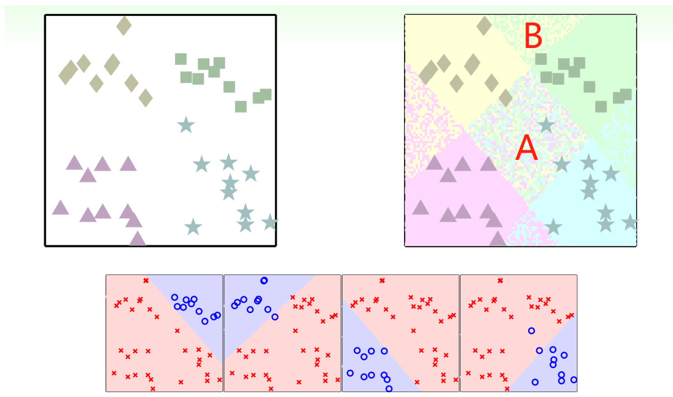

图 8: Ties

上图中，落在A区域的测试样本会被4个分类器同时拒绝，而落在B区域的测试样本会被同时判别为菱形与正方形。

一种修正方法是，使用logistic regression训练每一个分类器。这样，对于某个测试样本，我们可以用每个logistic分类器计算它所属该类别的“概率”，然后以概率最大的那个类别作为预测值。

上述解决多元分类问题的方法被称为“One-Versus-All”，简称OVA。简单来说包含两步：第一步，对于每个类别$k \in \mathcal{Y}$，在数据集：

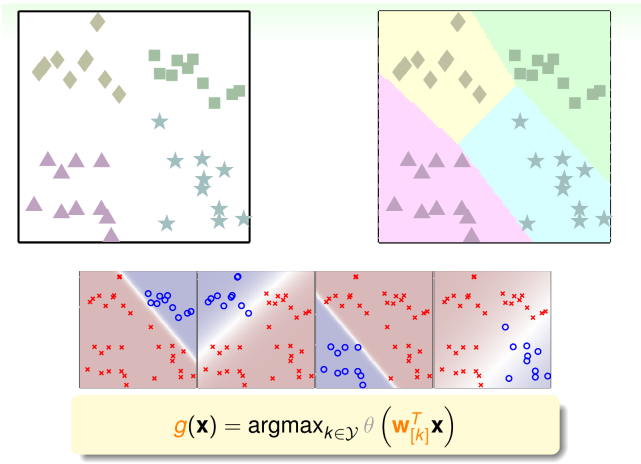

图 9: OVA-Logistic regression

$$\mathcal{D}_{[k]} = \left\{\left(\mathbf{x}_n, y_n^{'} = 2I(y_n = k) - 1\right)\right\}_{n=1}^{N}$$

上跑logistic regression，得到分类器$\mathbf{w}_{[k]}$；第二步，结合所有分类器，返回总分类器：

$$g(\mathbf{x}) = \arg\max_{k\in\mathcal{Y}} \left(\mathbf{w}_{[k]}^T\mathbf{x}\right)$$

OVA的优点是效率高，要计算的分类器个数也不算太多；缺点是当K很大，我们在训练每个分类器时都在面临unbalanced数据的问题。

4. Multiclass via Binary Classification

为了避免在unbalanced数据上进行训练，考虑“One-Versus-One”方法，或简称OVO法。该算法每次将两个类别的样本拿出来做一次二元分类，最终一共得到C_k^2个分类器。对于测试样本，我们用这C_k^2个分类器对它都预测一遍，看哪个类别被预测的次数最多——类似于循环积分赛。

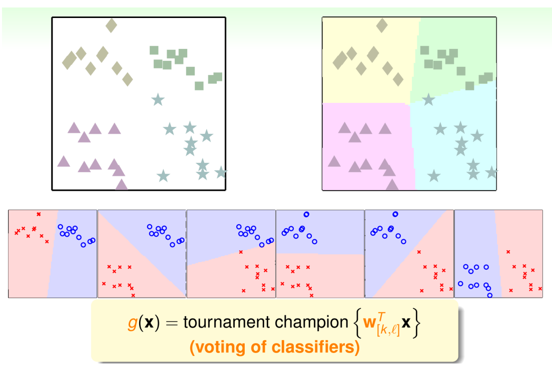

图 10: OVO

总的来说，OVO算法包括两步：第一步，对于每两个类别$(k,l) \in \mathcal{Y} \times \mathcal{Y}$，在数据集：

$$\mathcal{D}_{[k,l]} = \left\{ (\mathbf{x}_n, y_n^{'} = 2I(y_n = k) - 1): \quad y_n = k \ or \ y_n = l \right\}$$

上跑linear binary classification，得到分类器$\mathbf{w}_{[k,l]}$；第二步，结合所有分类器，返回总分类器：

$$g(\mathbf{x}) = tournament\ champion \quad \left(\mathbf{w}_{[k,l]}^T \mathbf{x} \right)$$

OVO的优点是避免了unbalanced data的问题，且在小规模学习问题上效率较好；缺点是，需要更多的空间，训练更慢，预测也慢。