Classification

问题：input 一只宝可梦，output 它的 type

输入数值化

对于宝可梦的分类问题来说，我们需要解决的第一个问题就是，怎么把某一只宝可梦当作 function 的 input

将宝可梦的特性数值化组成 vector 来描述它

Classification as Regression?

Regression 定义 model 好坏的定义方式对 classification 来说不适用；Regression 的 output 是连续性数值，而 classification 要求 output 是离散型的数值，我们很难找打一个 Regression 的 function 使大部分样本点的 output 都集中在某几个离散点的附近

注意：如果是多元分类问题，把 class1 的 target 当作 1，class2 的 target 当作 2，class3 的 target 当作 3 的做法是错误的；因为这种做法，就会被 Regression 认为 class1 与 class2 关系比较近，class2 与 class3 关系比较近；而 class1 与 class3 的关系比较远；但是当这些 class 之间并没有什么特殊的关系时，这样的标签用 Regression 是没有办法得到好的结果的（可以考虑 one-hot 编码作为解决方案）

Function(Model)

我们要找的 function f(x)里面会有另外一个 function g(x),当我们 input x 输入后,如果 g(x)>0,那 f(x)的输出就是 class 1，如果 g(x)<0,那 f(x)的输出就是 class 2，这个方法保证了 function 的 output 都是离散的表示 class 的数值

Loss function

我们可以把 loss function 定义成$L(f)=\sum_n \delta(f(x^n)\neq \hat{y}^n)$,即这个 model 在所有的 training data 上 predict 预测错误的次数，也就是说分类错误的次数越少，这个 function 表现得越好

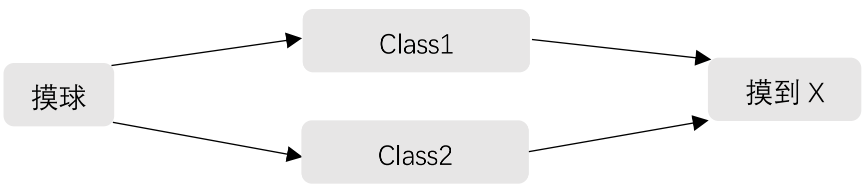
但是这个 loss function 没办法微分，是无法用 gradient descent 的方法去解的，当然有 perceptron、SVM 这些方法可以用，但这里先用另一个 solution 来解决这个问题

Solution：Generative model

概率理论解释

假设我们考虑一个二元分类的问题，我们拿到一个 input x,想要知道这个 x 属于 class1 或 class2 的概率实际上就是一个贝叶斯公式，x 属于 class1 的概率就等于 class1 自身发生的概率乘上在 class1 里取出 x 这种颜色的球的概率除以在 class1 和 class2 里取出 x 这种颜色的球的概率(后者是全概率公式)

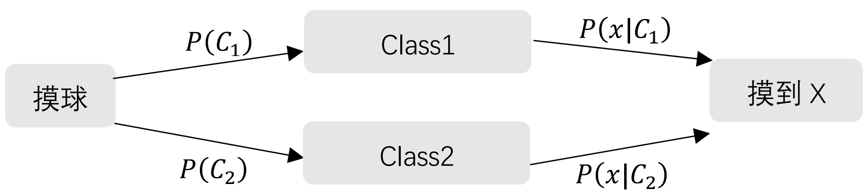
贝叶斯公式=单条路径概率/所有路径概率之和



因此我们想要知道x属于 class1 或是 class2 的概率，只需要知道 4 个值：

$P(C_1)$，$P(x|C_1)$，$P(C_2)$，$P(x|C_2)$，我们希望从 training data 中估测出这四个值

流程图简化如下：



于是我们得到:(分母为全概率公式)

- x属于 class1 的概率为第一条路径除以两条路径和：$P(C_1|x)=\frac{P(C_1)P(x|C_1)}{P(C_1)P(x|C_1)+P(C_2)P(x|C_2)}$
- x属于 class2 的概率为第二条路径除以两条路径和：$P(C_2|x)=\frac{P(C_2)P(x|C_2)}{P(C_1)P(x|C_1)+P(C_2)P(x|C_2)}$

这一整套叫做 Generative model(生成模型)，为什么叫它 Generative model 呢？因为有这个 model 的话，就可以

拿它来 generate 生成x(如果你可以计算出每一个x出现的概率,就可以用这个 distribution 分布生成x、sample$x$*出来*)

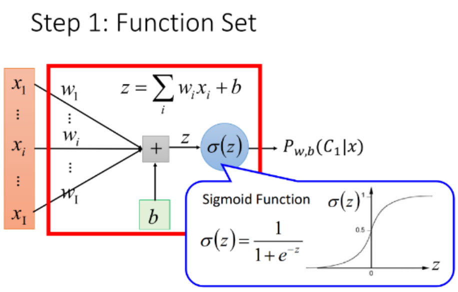
Three Steps of classification
现在我们来回顾一下做 classification 的三个步骤，实际上也就是做 machine learning 的三个步骤

- Find a function set(model)
  这些 required probability$P(C)$和 probability distribution$P(x|C)$就是 model 的参数，选择不同的 Probability distribution(比如不同的分布函数，或者是不同参数的 Gaussian distribution)，就会得到不同的 function，把这些不同参数的 Gaussian distribution 集合起来，就是一个 model，如果不适用高斯函数而选择其他分布函数，就是一个新的 model 了
  当这个 posterior Probability $P(C|x)>0.5$ 的话，就 output class1，反之就 output class2($P(C_1|x)+P(C_2|x)=1$)，因此没必要对 class2 在计算一遍）
- Goodness of function
  对于 Gaussian distribution 这个 model 来说，我们要评价的是决定这个高斯函数形状的均值u和协方差C这两个参数的好坏，而极大似然函数$L\left(u, C\right)$的输出值，就评价了这组参数好坏
- Find the best function
  找到的那个最好的 function，就是使$L\left(u, C\right)$值最大的那组参数，实际上就是所有样本点的均值和协方差

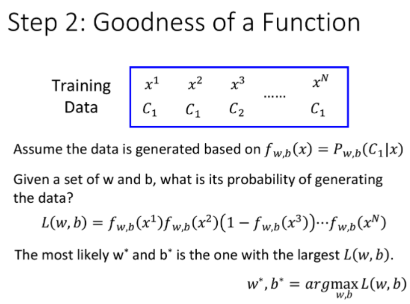
$$u^*=\frac{1}{n}\sum_{i=0}^{n}x^i \quad C^*=\frac{1}{n}\sum_{i=0}^{n}(x^i-u^*)\left(x^i-u^*\right)^T$$

  这里上标i表示第i个点，这里x是一个 features 的 vector，用下标来表示这个 vector 中的某个 feature

Logistic Regression
Step1：function set
这里的 function set 就是 Logistic Regression----逻辑回归
w_i：weight，b:bias,$\sigma(z)$：sigmoid function，x_i：input



Step2：Goodness of a function
现在我们有 N 笔 Training data，每一笔 data 都要标注它是属于哪一个 class
假设这些 Training data 是从我们定义的 posterior Probability，那我们就可以去计算某一组w和b去产生这 N 笔 Training data 的概率，利用极大似然估计的思想，最好的那组参数就是最大可能性产生当前 N 笔 Training data 分布的w^*和b^*
似然函数只需要将每一个点产生的概率相乘即可，注意，这里假定是二元分类，class2 的概率为 1 减去 class1 的概率



由于$L(w,b)$是乘积项的形式，为了方便计算，我们将上式做个变换:

$$w^*, b^* = argmaxL(w,b) = argmin\big(-lnL(w,b)\big)$$

$$-lnL(w,b) = -lnf_{w,b}(x^1) - lnf_{w,b}(x^2) - ln(1 - f_{w,b}(x^3)) \cdots\cdots$$

由于 class1 和 class2 的概率表达式不统一，上面的式子无法写成统一的形式，为了统一格式，这里 Logistic Regression 里的所有 Training data 都打上 0 和 1 的标签，即 output $\hat{y} = 1$代表 class1，output $\hat{y} = 0$代表 class2

于是上式进一步改写成:

$$-lnL(w,b) = -\sum_n [\hat{y}^i ln\, f_{w,b}(x^i) + (1-\hat{y}^i)ln(1 - f_{w,b}(x^i))]$$

假设有两个 distribution p 和 q,它们的交叉熵就是$H(p,q) = -\sum_x p(x)\ln\,(q(x))$

Cross entropy 交叉熵的含义是表达这两个 distribution 有多接近，如果 p 和 q 这两个 distribution 一模一样的话，那它们算出来的 cross entropy 就是 0,而这里$f(x^n)$表示 function 的 output，$\hat{y}^i$表示预期的 target,因此交叉熵实际上表达的是希望这个 function 的 output 和它的 target 越接近越好

总之，我们要找的参数实际上就是:

$$w^*, b^* = argmaxL(w,b) = argmin\big(-lnL(w,b)\big) = -\sum_n [\hat{y}^i ln\, f_{w,b}(x^i) + (1-\hat{y}^i)ln(1 - f_{w,b}(x^i))]$$

Step3：Find the best function

实际上就是去找到使 loss function 即交叉熵之和最小的那组参数w^*, b^*就行了，这里用 gradient descent 的方法进行运算就 ok

这里 sigmoid function 的微分可以直接作为公式记下来：$\frac{\partial\sigma(z)}{\partial z} = \sigma(z)\big(1-\sigma(z)\big)$,sigmoid 和它的微分的图像如下：

$$\frac{\partial -lnL(w,b)}{\partial w_i} = -\sum_n[\hat{y}^i\frac{\partial lnf_{w,b}(x^i)}{\partial w_i} + (1-\hat{y}^i)\frac{\partial ln(1-f_{w,b}(x^i))}{\partial w_i}]$$

$$\frac{\partial lnf_{w,b}(x^i)}{\partial w_i} = \frac{\partial lnf_{w,b}(x)}{\partial z}\frac{\partial z}{\partial w_i}, \frac{\partial lnf_{w,b}(x)}{\partial z} = \frac{\partial ln\sigma(z)}{\partial z} = \frac{1}{\sigma(z)}\sigma(z)\big(1-\sigma(z)\big) = \big(1-\sigma(z)\big), \frac{\partial z}{\partial w_i} = x_i$$

$$\frac{\partial ln(1-f_{w,b}(x^i))}{\partial w_i} = \frac{\partial ln\big(1-\sigma(z)\big)}{\partial z}\frac{\partial z}{\partial w_i}, \frac{\partial ln\big(1-\sigma(z)\big)}{\partial z} = \frac{-1}{1-\sigma(z)}\sigma(z)\big(1-\sigma(z)\big) = -\sigma(z), \frac{\partial z}{\partial w_i} = x_i$$

$$\frac{\partial - lnL(w,b)}{\partial w_i} = -\sum_n [\hat{y}^i\left(1-f_{w,b}(x^i)\right)x_i + (1-\hat{y}^i) - f_{w,b}(x^i)x_i]$$

$$= -\sum_n(\hat{y}^i - f_{w,b}(x^i))x_i$$

$$w_i = w_i - \eta\sum_n -(\hat{y}^n - f_{w,b}(x^n))x_i^n$$

Logistic Regression vs Linear Regression

Compare in step1

Logistic Regression 是把每一个 featuresx_i加权求和，加上 bias，再通过 sigmoid function，当作 function 的 output

因为 Logistic Regression 的 output 是通过 sigmoid function 产生的，因此一定是介于 0~1 之间；而 Linear regression 的 output 并没有通过 sigmoid function，所以它可以是任何值

Compare in step2

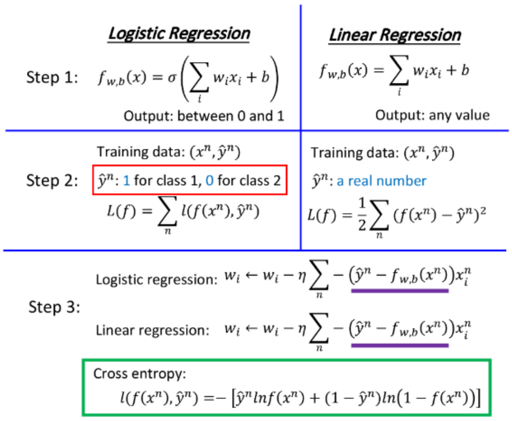
在 Logistic Regression 中，我们定义的 loss function，既要去 minimize 的对象，是所有 example 的 output($f(x^n)$)和实际 target($\hat{y}^n$)在伯努利分布下的 cross entropy 总和

交叉熵的描述：这里把$f(x^n)$和$\hat{y}^n$各自看作是一个两点分布，那它们的 cross entropy:$l(f(x^n),\hat{y}^n)=-\left[\hat{y}^n lnf(x^n)+(1-\hat{y}^n)ln\left(1-f(x^n)\right)\right]$之和，就是我们要 minimize 的对象，直观来讲，就是希望 function 的 output $f(x^n)$和它的 target$\hat{y}^n$越接近越好

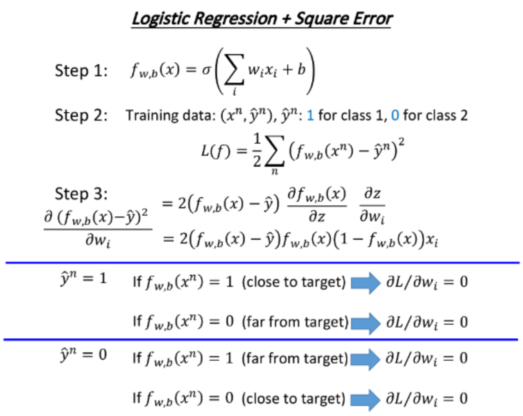
而在 Linear Regression 中，loss function 的定义相对简单，就是单纯的 function 的 output($f(x^n)$)和实际 target($\hat{y}^n$)在数值上的平方和的均值

Compare in step3

Surprising，Logistic Regression 和 Linear Regression 的$w_i$update 的方式一模一样，唯一区别在于，Logistic Regression 的 target($\hat{y}^n$)和 output $f(x^n)$都必须是在 0 和 1 之间的，而 Linear Regression 的 target 和 output 的范围可以是任意值



Logistic Regression and Square error?



现在会遇到一个问题:如果第 n 个点的目标 target 是 class1，则$\hat{y}^n=1$，此时如果 function 的 output $f(x^n)=1$的话，说明现在离 target 很近了，$f(x^n)-\hat{y}^n$这一项是 0，于是得到的微分$\frac{\partial L}{\partial w_i}$会变成 0，当然这种情况是合理的；但是当 function 的 output $f(x^n)=0$的时候，说明 target 还很遥远，但是由于在 step3 中求出来的 update 表达式中有一个$f(x^n)$，因此此时得到的微分$\frac{\partial L}{\partial w_i}$会变成 0

Conclusion 结论

对于分类的问题(主要是二元分类)，我们一般有两种方法去处理问题，一种是 Generative 的方法，另一种是 Discriminative 的方法，注意到分类问题的 model 都是从贝叶斯方差出发的，即

$$P(C_i|x)=\frac{P(C_i)P(x|C_i)}{\sum_{j=1}^{n}P\left(C_j\right)P(x|C_j)}=\sigma(z)=\frac{1}{1+e^{-z}}=\frac{1}{1+e^{-(b+\sum_k w_k x_k)}}$$

其中分子表示属于第i类的可能性，分母表示遍历从 1 到 n 所有的类的可能性，这两种方法的区别在于 Generative model 会假设一个带参数的 Probability contribute，利用这个假设的概率分布函数计算$P(x|C_i)$和$P\left(x|C_j\right)$,结合极大似然估计法最终得到最优的参数以确定这个 model 的具体形式

Discriminative model 不作任何假设，因此它无法通过假定的 Probability distribution 得到$P(x|C_i)$的表达式，直接利用交叉熵和 gradient descent 结合极大似然估计法得到最优w和b，以确定 model 的具体形式，最后，利用得到的$P(C_i|x)$与 0.5 相比较来判断它属于那个 class 的可能性更大

Generative model and Discriminative model 的优势对比

Generative model 的优势：它对 data 的依耐性并没有像 Discriminative model 那么严重，在 data 数量少或者 data 本身就存在 noise 的情况下受到的影响较小，而且它还可以做到 Prior 部分和 class-dependent 部分分开处理，如果可以借助其他方式提高 Prior model 的准确率，对整一个 model 是有所帮助的(比如前面提到的语音辨识)

Discriminative model 优势：在 data 充足的情况下，它训练出来的 model 的准确率一般比 Generative model 要来的高

Multi-class Classification

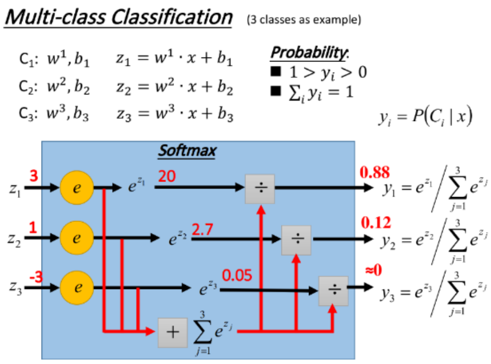
softmax

之前讲的都是二元分类的情况，这里讨论一下多元分类问题，其原理的推导过程与二元分类基本一致，假设有三个 class：C_1,C_2,C_3,每一个 class 都有自己的 weight 和 bias，这里w_1,w_2,w_3分布代表三个 vector，b_1,b_2,b_3分别代表三个 const，input x也是一个 vector

Softmax 的意思是对最大值做强化，因为在做第一步的时候，对 z 取 exponential 会使大的值和小的值之间的差距被拉的更开，也就是强化大的值

我们把z_1,z_2,z_3丢进一个 softmax 的 function，softmax 做的事情是如下三步：

- 取 exponential，得到e^{z_1},e^{z_2},e^{z_3}
- 把三个 exponential 累计求和，得到 total sum=$\sum_{j=1}^{3} e^{z_1}$
- 将 total sum 分别除去去这三项(归一化)，得到$y_1=\frac{e^{z_1}}{\sum_{j=1}^{3} e^{z_1}}$、$y_2=\frac{e^{z_2}}{\sum_{j=1}^{3} e^{z_1}}$、$y_3=\frac{e^{z_3}}{\sum_{j=1}^{3} e^{z_1}}$



原来的 output z 可以是任何值，但是做完 softmax 之后，你的 output y_i的值一定是介于 0~1 之间，并且它们的和一定是 1，$\sum_i y_i=1$，以上图为例，y_i表示 input x属于第i个 class 的概率，比如属于C_1的概率是$y_i=0.88$，属于C_2的概率是$y_i=0.12$，属于C_3的概率是$y_i=0$

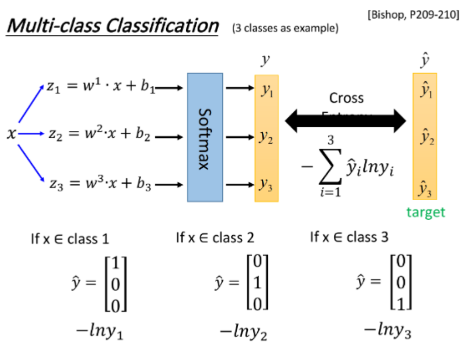
而 softmax 的 output，就是拿来当z的 posterior probability

假设我们用的是 Gaussian distribution(共用 covariance)，经过一般推导以后可以得到 softmax function，而从 information theory 也可以推导出 softmax function，Maximum entropy 本质内容和 Logistic Regression 是一样的，它是从另一个观点来切入为什么我们的 classification 长这样子

Muti-class classification 的过程:

如下图所示，input x经过三个式子分别生成z_1,z_2,z_3,经过 softmax 转化成 output y_1,y_2,y_3,它们分别这三个 class 的 posterior probability，由于 summation=1，因此做完 softmax 之后就可以把 y 的分布当做是一个 probability contribution，我们在训练的时候还需要有一个 target，因为是三个 class，output 是三维的，对应的 target 也是三维的，为了满足交叉熵的条件，target $\hat{y}$也必须是 probability distribution，这里我们不能使用 1,2,3 作为 class 的区分，为了保证所有 class 之间的关系是一样的，这里使用类似于 one-hot 编码的方式，即

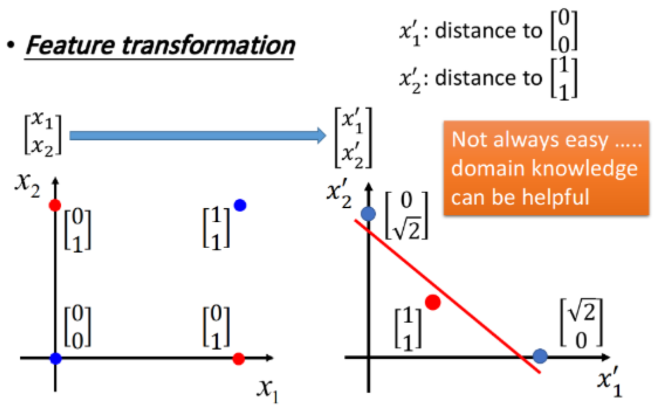
$$\hat{y}=\begin{bmatrix}1\\0\\0\end{bmatrix}_{x\in class1} \quad \hat{y}=\begin{bmatrix}0\\1\\0\end{bmatrix}_{x\in class2} \quad \hat{y}=\begin{bmatrix}0\\0\\1\end{bmatrix}_{x\in class3}$$



这个时候就可以计算一下 output y和 target $\hat{y}$之间的交叉熵，即$-\sum_{i=1}^{3}\hat{y}_i lny_i$，同二元分类一样，多元分类问题也是通过极大似然估计法得到最终的交叉熵表达式

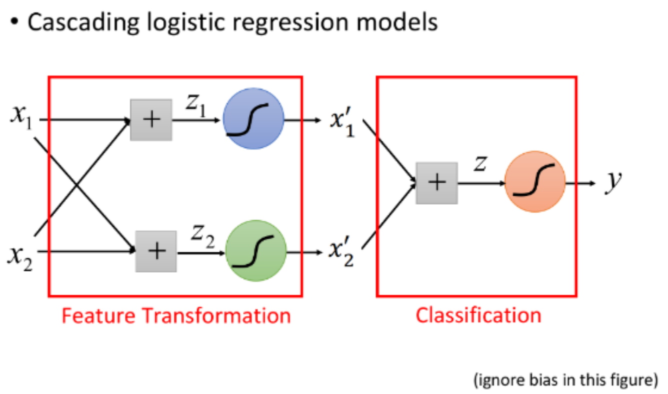
Feature Transformation

如果坚持要用 Logistic Regression 的话，有一招叫做 Feature Transformation，原来的 feature 分布不好划分，那我们可以将之转化以后，找一个比较好的 feature space，让 Logistic Regression 能够处理假设这里定义x_1'是原来的点到$\begin{bmatrix}0\\0\end{bmatrix}$之间的距离，$x_2'$是原来的点到$\begin{bmatrix}1\\1\end{bmatrix}$之间的距离，重新映射之后如下图所示(红色两个点重合)，此时 Logistic Regression 就可以把它们划分开来



但麻烦的是，我们并不知道怎么做 feature Transformation，如果在这上面花费太多的时间就得不偿失了，于是我们会希望这个 Transformation 是机器自己产生的，怎么让机器自己产生呢？我们可以让很多的 Logistic regression 连接起来

我们让一个 input x的两个 feature x_1,x_2经过两个 Logistic Regression 的 transform，得到新的 feature x_1',x_2',在这个新的 feature space 上，class1 和 class2 是可以用一条直线分开的，那么最后只要再接另外一个 Logistic Regression 的 model，它根据新的 feature，就可以把 class1 和 class2 分开



因此着整个流程是，先用 n 个 Logistic Regression 做 feature Transformation(n 为每个样本点 feature 数量)，生成 n 个新的 feature，然后再用一个 Logistic regression 做 classifier

Logistic regression 的 boundary 一定是一条直线，它可以有任意画法，但肯定是按照某个方向从高到低等高线分布，具体的分布是由 Logistic regression 的参数决定的，每一条直线都是由$z=b+\sum_i^n w_i x_i$组成的(二维 feature 的直线画在二维平面上，多维 feature 的直线则是画在多维空间上)

注意，这里的 Logistic regression 只是一条直线，它指的是“属于这个类”或“不属于这个类”这两种情况，因此最后的这个 Logistic regression 是跟要检测的目标类相关的，当只是二元分类的时候，最后只需要一个 Logistic regression 即可，当面对多元分类问题，需要用到多个 Logistic regression 来画出多条直线画出所有的类，每一个 Logistic regression 对应它要检测的那个类