ML Note

qhy

2017 年 7 月 31 日

目录

1 SVM

定义标签:$\hat{y} = \{+1, -1\}$,+1表示正类,-1表示负类定义输出:

$$g(x) = \begin{cases} +1 & , f(x) > 0 \\ -1 & , f(x) > 1 \end{cases} \tag{1}$$

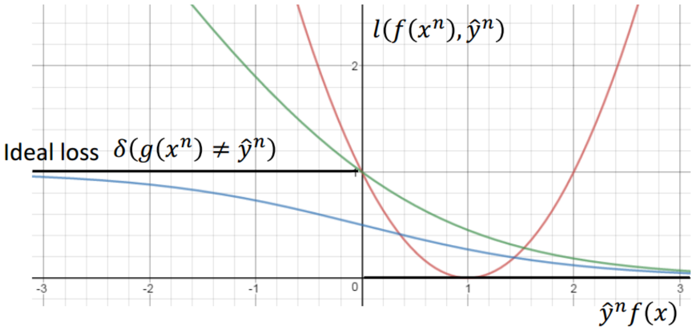
定义loss functon:

$$L(f) = \sum_n I(g(x^n \neq \hat{y}^n)) \tag{2}$$

也就是统计分类错误的个数,从更一般的角度来看,就是每分类错误一个,就给一定的惩罚

分析:$\hat{y^n} f(x^n) > 0$的时候,分类正确,无论是正类还是负类,

如果把这个看做一个新的变量t,那么损失函数(一个数据的损失函数)就可以定义成$t > 0$时无惩罚,$t < 0$的时候给出一定的惩罚。



理想的情况就是图1中的黑线,分类错误就给出惩罚。

由于此loss function使用梯度下降算法麻烦,故使用一个惩罚函数l代替I:

$$L(f) = \sum_n I(f(x^n, \hat{y}^n)) \tag{3}$$

红色线为**Square loss**:

$$l(f(x^n, \hat{y}^n) = (\hat{y}^n f(x^n) - 1)^2 \tag{4}$$

显然是不合理的,因为它对于分类正确的结果也给出了惩罚。

蓝色线为**sigmoid+ square loss**:

$$l(f(x^n, \hat{y}^n) = (\sigma(\hat{y}^n f(x^n)) - 1)^2 \tag{5}$$

这样,得到的结果是合理的,对于分类正确,确定性高的给出低惩罚,对于分类错误,甚至错误的离谱的,则给出大的惩罚

但是有一个问题就是,对于分类错误的结果,它的惩罚很低。

绿色线为**Sigmoid + cross entropy**

$$l(f(x^n, \hat{y}^n) = \ln(1 + exp(-y^n f(x^n))) \tag{6}$$

蓝色线为**hinge loss**:

$$l(f(x^n, \hat{y}^n) = max(0, 1 - y^n f(x^n)) \tag{7}$$

对于分类正确的结果,只要$yf(x) < 1$的数据还是会给出惩罚,而SVM采用的就是$HingeLoss$

从$f(x)$来看,$f(x)$为分界线,只要在$f(x)$上下距离为1的样本都会给出惩罚。

1.1 线性SVM

模型如下:

- Step 1: Function (Model)

New w

$$f(x)=\sum_i w_i x_i + b = \begin{bmatrix} w \\ b \end{bmatrix} \cdot \begin{bmatrix} x \\ 1 \end{bmatrix} = w^T x$$

New x

- Step 2: Loss function

$$L(f)=\sum_n l(f(x^n),\hat{y}^n) + \lambda \|w\|_2$$

convex

$$l(f(x^n),\hat{y}^n) = max\left(0, 1-\hat{y}^n f(x)\right)$$

梯度下降:

$$\frac{\partial L(f)}{\partial w_i} = \sum_n \underbrace{-\delta(\hat{y}^n f(x^n) < 1)\hat{y}^n x_i}_{c^n(w)} \qquad w_i \leftarrow w_i - \eta \sum_n c^n(w) x_i^n$$

1.2 线性SVM2

至此,线性SVM结束,但是这个SVM和之前见到的SVM貌似不太一样?对偶问题,SMO,等等这些东东呢?

另一种描述:

Minimizing loss function L:

$$L(f)=\sum_n \varepsilon^n + \lambda \|w\|_2$$

$$\boxed{\varepsilon^n = max\left(0, 1-\hat{y}^n f(x)\right)}$$

ε^n: slack variable

Quadradic programming problem

$$\boxed{\begin{array}{l} \varepsilon^n \geq 0 \\ \varepsilon^n \geq 1-\hat{y}^n f(x) \Rightarrow \hat{y}^n f(x) \geq 1-\varepsilon^n \end{array}}$$

经过这样处理,原问题从最小化:

$$L(f)=\sum_n l(f(x^n),\hat{y}^n) + \lambda \|w\|_2$$

变成了最小化:

$$L(f)=\sum_n \varepsilon^n + \lambda \|w\|_2 \quad \text{s.t.} \quad \hat{y}^n f(x) \geq 1-\varepsilon^n$$

进一步处理,问题变成最小化:

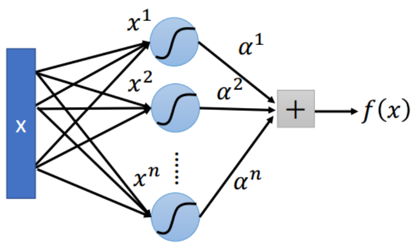
$$L(f)= \lambda \|w\|_2 \quad \text{s.t.} \quad \hat{y}^n f(x) \geq 1$$

1.3 Kernel SVM

简单地从梯度下降的角度来看,那么最终得到的w^*一定是所有x_i的线性组合,i.e.

$$w^* = \sum_i a_i^* x_i \tag{8}$$

而由于使用**hinge loss**的原因,大部分的$a_i^* = 0$,对于$a_i^* \neq 0$的x_i,则称为**Support Vector**
那么最终的解的形式变成了$f(x) = \sum_i a_i (x_i \cdot x)$
若引入核函数,则是$f(x) = \sum_i a_i \cdot K(x_i \cdot x)$
而这个解的形式,则可以看成是一个单隐层神经网络:



其中,每一个**Support Vector**表示一个神经元。
这里为了引入核函数,把问题变成了这种形式,参数变成了a,但一样可以用,梯度下降去解决,
从这里来看,应该是由a决定哪些是 **支持向量**,而不是由支持向量,决定哪些$a = 0$
为什么说RBF是对应的是无穷为的点积?
因为RBF使用泰勒展开之后,有无穷项,每一项代表若干个维度的分量分别相乘之和的结果。
怎么设计核函数? 满足 mercer 条件的任一函数都可以是核函数。

1.4 SVM的应用与变种

- SVR
- Ranking SVM
- one-class SVM
- Recursive Kernel

2 Structure Learning

之前我们接触的大部分都是输入是一个向量,输出是一个实数但如果我们要输出一个序列则怎么处理?

这里有一个大致的框架:
设X是输入,Y是输出,$F(X,Y)$描述X,Y的相似度,或者是X输出Y的概率
对于训练过程,我们要做的就是最大化训练样本输入和输出的相似度,i.e.

$$\max F(X,Y) \tag{9}$$

对于测试过程,我们的输出就是在Y的空间中,与测试输入X,$F(X,Y)$最大的那个Y
那么,就有3个问题需要解决:

1. $F(X,Y)$是什么样子?
2. 怎么有效快速地输出y?
3. 怎么训练?

总的来说,就是把X和Y两个不同的东西,表示在同一个空间内,这样才能建立起度量,进而才有优化的方向。

这三个问题和 HMM中的三个问题是一致的,因为HMM就是Structure Learning中的一种

2.1 Structured Liner Model

F(x,y)的形式:y可以看成是x在特征空间中的投影,我们要做的就是在特征空间中的点距离y尽可能地近,或者是相似度尽可能大。

$$\mathrm{F}(x,y)=\underbrace{\begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \end{bmatrix}}_{w}\cdot\underbrace{\begin{bmatrix} \phi_1(x,y) \\ \phi_2(x,y) \\ \phi_3(x,y) \\ \vdots \end{bmatrix}}_{\phi(x,y)}$$

$$\Downarrow$$

$$\mathrm{F}(x,y)=w\cdot\phi(x,y)$$

我们可以简单地定义损失函数就是$F(x,\hat{y})-F(x,d\dot{y})$,也就是当前预测的点到目标点的距离。
其中$\dot{y}$是Fx,y最大的y,我们只需要考虑最大的y即可,而不需要考虑其他。
至此,我们就可以用梯度下降的方法进行训练:

- **Input**: training data set $\{(x^1,\hat{y}^1),(x^2,\hat{y}^2),\ldots,(x^r,\hat{y}^r),\ldots\}$
- **Output**: weight vector w
- **Algorithm**: Initialize w = 0
  - do
    - For each pair of training example $(x^r,\hat{y}^r)$
      - Find the label $\tilde{y}^r$ maximizing $w\cdot\phi(x^r,y)$

        $\tilde{y}^r=\arg\max_{y\in Y} w\cdot\phi(x^r,y)$ (question 2)
      - If $\tilde{y}^r\neq\hat{y}^r$, update w

        $w\rightarrow w+\phi(x^r,\hat{y}^r)-\phi(x^r,\tilde{y}^r)$
  - until w is not updated $\Rightarrow$ We are done!

收敛性证明:对于线性可分的数据,我们可以证明,最多需要$(\frac{R}{\delta})^2$次迭代就可以收敛。
其中,R表示特征空间中任意两个y之间的最大距离,δ表示margin
也就是说,迭代的次数和样本的规模无关。

$$w^k = w^{k-1} + \phi(x^n, \hat{y}^n) - \phi(x^n, \tilde{y}^n) \quad \text{(the relation of } w^k \text{ and } w^{k-1}\text{)}$$

Proof that: The angle ρ_k between $\hat{w}$ and w_k is smaller as k increases

i.e.
Analysis $\cos\rho_k$

$$\cos\rho_k = \boxed{\frac{\hat{w}}{\|\hat{w}\|} \cdot \frac{w^k}{\|w^k\|}}$$

$$\hat{w}\cdot w^k = \hat{w}\cdot\left(w^{k-1} + \phi(x^n, \hat{y}^n) - \phi(x^n, \tilde{y}^n)\right)$$

$$= \hat{w}\cdot w^{k-1} + \underbrace{\hat{w}\cdot\phi(x^n, \hat{y}^n) - \hat{w}\cdot\phi(x^n, \tilde{y}^n)}_{\geq\delta\ \text{(Separable)}} \geq \hat{w}\cdot w^{k-1} + \delta$$

即,我们要证明,随着迭代的过程,w是不断向最优解靠近的,也就是夹角不断变小

$$\hat{w}\cdot w^k \geq \hat{w}\cdot w^{k-1} + \delta \longrightarrow \hat{w}\cdot w^k \geq k\delta$$

$$\|w^k\|^2 = \|w^{k-1} + \phi(x^n, \hat{y}^n) - \phi(x^n, \tilde{y}^n)\|^2$$

$$= \|w^{k-1}\|^2 + \underbrace{\|\phi(x^n, \hat{y}^n) - \phi(x^n, \tilde{y}^n)\|^2}_{>0} + \underbrace{2w^{k-1}\cdot\left(\phi(x^n, \hat{y}^n) - \phi(x^n, \tilde{y}^n)\right)}_{?\ <0\ \text{(mistake)}}$$

Assume the distance between any two feature vector is smaller than R

$$\leq \|w^{k-1}\| + R^2$$

$$\|w^1\|^2 \leq \|w^0\|^2 + R^2 = R^2$$

$$\|w^2\|^2 \leq \|w^1\|^2 + R^2 \leq 2R^2$$

$$\cdots$$

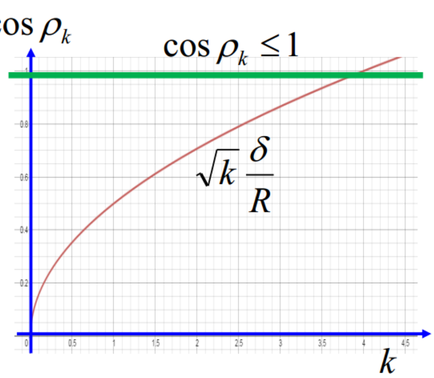
$$\boxed{\|w^k\|^2 \leq kR^2}$$

$$\cos\rho_k = \frac{\hat{w}}{\|\hat{w}\|} \cdot \frac{w^k}{\|w^k\|} \qquad \hat{w}\cdot w^k \geq k\delta \qquad \|w^k\|^2 \leq kR^2$$

$$\geq \frac{k\delta}{\sqrt{kR^2}} = \sqrt{k}\,\frac{\delta}{R}$$

$$\sqrt{k}\,\frac{\delta}{R} \leq 1$$

$$k \leq \left(\frac{R}{\delta}\right)^2$$



这里我们并没有证明$\cos\rho$是增大到1的,我们证明的是它的一个下界是不断增大到1,那么根据夹逼定理得证。

值得注意的是,这里有一个参数δ,这个需要实现认为设定,当约束比较严格的是,则迭代次数较多,当约束比较松的时候,则迭代次数少。

δ:对于这个参数的理解,我们并不要求整个算法的输入X在特征空间中的表示一定要和Y的表示重合,只要它的差值在一定的范围内都是可接受的,因此,这个参数也叫margin