文本分类中的损失函数分析

你的姓名学号

论文中的文本分类方法损失函数分析

1. 支持向量机（SVM）

- **损失函数:** $L_c = \sum_{i=1}^{n}(1 - y_i\vec{x}_i\vec{\beta})_+ + \lambda\|\vec{\beta}\|^2$
- **特点:**
  - 训练损失：合页损失 $(1 - y_i f(\vec{x}_i))_+$
  - 仅对间隔内的样本 $(y_i f(\vec{x}_i) \leq 1)$ 进行惩罚
  - 正则项：用于最大间隔优化的 L2 范数
  - svm 是基于结构化风险最小化的模型，使用间隔的倒数作为正则项可以保证在训练的过程中尽可能使得间隔倒数减小，以此保证间隔最大，而间隔越大，有新数据加入的时候分类正确的可能性越大，以此减小了过拟合的可能性，除此之外，svm 还引入了软间隔的概念，牺牲部分训练时的准确性来保证结构风险最小，这种策略如果选取参数合适可以很好地去除训练数据集中的噪音，使预测的准确性大大上升
- **区别:** 唯一使用非光滑合页损失的方法，忽略间隔外正确分类的样本

2. 线性最小二乘拟合（LLSF）

- **损失函数:** $L_c = \sum_{i=1}^{n}(1 - y_i\vec{x}_i\vec{\beta})^2 + \lambda\|\vec{\beta}\|^2$
- **特点:**
  - 训练损失：二次损失（平方误差）
  - 由于二次项的存在，对异常值惩罚较强
  - 正则项：L2 范数（岭回归）
- **区别:** 唯一对“过度正确”样本 $(y_i f(\vec{x}_i) \gg 1)$ 进行惩罚的方法

3. 逻辑回归（LR）

- **损失函数:** $L_c = \sum_{i=1}^{n}\log(1 + \exp(-y_i\vec{x}_i\vec{\beta})) + \lambda\|\vec{\beta}\|^2$
  $\mathcal{L}_{\mathrm{CE}} = -\frac{1}{N}\sum_{i=1}^{N}\Big[y_i\log(p_i) + (1 - y_i)\log(1 - p_i)\Big] + \lambda\|\vec{\beta}\|^2$
- **特点:**
  - 逻辑回归在论文中和课程中用的损失函数的形式不同，论文中所用损失函数的正负例标记为 {-1，1}，而课程中学到的交叉熵损失函数正负例标记为 {0，1}，使用变量映射 y=2x-1 将 {0，1} 映射到 {-1，1} 可证明两种损失函数的本质相同

- sigmoid 函数将输出结果映射到 {0，1} 区间上，本质上是使用概率判断输出结果是正类还是负类（sigmoid 函数可看成分类的概率，大于 0.5 为正类，小于 0.5 为负类）
- 正则项：L2 范数

- **区别:** 概率解释是其与其他方法的主要区别

4. 神经网络

- **损失函数:** $L_c = \sum_{i=1}^{n}(1-\pi(y_i\vec{x}_i\vec{\beta}))^2 + \lambda\|\vec{\beta}\|^2$
  $-\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}\left[y_k^{(i)}\log\left((h_\Theta(x^{(i)}))_k\right) + (1-y_k^{(i)})\log\left((1-h_\Theta(x^{(i)}))_k\right)\right] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_{ji}^{(l)})^2$
- **特点:**
  - 论文中神经网络的训练误差采用误差平方和的形式，而课程中教授的是交叉熵方式，两种损失函数的神经网络均有自己的用处
  - 由于误差平方和在求梯度后，梯度中会存在 h(x)(1-h(x)) 项，这会导致在模型输出趋于 0 或趋于 1 的时候会发生梯度丢失导致迭代速度慢，训练速度慢，而交叉熵求梯度后则不会出现有关 sigmoid 的函数项，故在分类问题上使用交叉熵更为合适（交叉熵输出解释为概率，这点也非常符合分类任务），但在回归问题上，使用误差平方和更为合适，因为其物理意义直观（距离度量），数学形式简单，是回归任务的标准损失。
  - 正则项：L2 范数
- **区别:** 与 LLSF 相比，对误分类样本的惩罚较轻

5. Rocchio 风格分类器

- **损失函数:** $L_c = -\sum_{y_i=1} y_i\vec{x}_i\vec{\beta} - \frac{bN_c}{N_{\neg c}}\sum_{y_i=-1} y_i\vec{x}_i\vec{\beta} + \frac{N_c}{2}\|\vec{\beta}\|^2$
- **特点:**
  - 显式对正负样本赋予不同权重
  - 参数 b 控制负样本的影响
  - 正则化强度与正样本数 N_c 成正比
- **区别:** 唯一显式区分正负样本权重的方法

6. 原型分类器

- **损失函数:** $L_c = -\sum_{y_i=1} y_i\vec{x}_i\vec{\beta} + \frac{N_c}{2}\|\vec{\beta}\|^2$
- **特点:**
  - 训练损失：仅优化正样本
  - 忽略负样本（损失 =0）
  - 正则项：由 N_c 缩放的 L2 范数
- **区别:** 与朴素贝叶斯（NB）共享训练损失，但使用不同正则项

7. K 近邻（kNN）

- **损失函数:** $L_c = -\sum_{\substack{y_i=1 \\ \vec{x}_i \in R_k(\vec{x})}} y_i \vec{x}_i \vec{\beta}_x + \frac{1}{2}\|\vec{\beta}_x\|^2$
- **特点:**
  - 局部损失函数（依赖测试样本 $\vec{x}$）
  - 仅考虑局部邻域 $R_k(\vec{x})$ 内的正样本
  - 正则项：应用于局部参数 $\vec{\beta}_x$ 的 L2 范数
- **区别:** 唯一非全局优化方法（损失依赖测试样本）

8. 朴素贝叶斯（NB）

- **无平滑:** $L_c = -\sum_{y_i=1} y_i \vec{x}_i \vec{\beta} + S_c \|e^{\vec{\beta}}\|_1$
- **拉普拉斯平滑:** $L_c = -\sum_{y_i=1} y_i \vec{x}_i \vec{\beta} + (p + S_c)\|e^{\vec{\beta}}\|_1 + \|\vec{\beta}\|_1$
- **特点:**
  - 训练损失：与原型分类器相同（仅正样本）
  - 独特的指数正则项 $\|e^{\vec{\beta}}\|_1$，其自动满足概率非负性，避免复杂约束；整体损失为凸函数，可高效求解；通过引入维度 p 直接编码拉普拉斯平滑的归一化补偿。拉普拉斯正则项的设计核心是解决特征零概率问题并控制模型复杂度。拉普拉斯平滑采用分子加一分母加 k（k 为类别数）的方法估计没有出现的现象的概率
  - 正则化强度依赖于 S_c（词频）和 p（特征维度）
  - 朴素贝叶斯采用贝叶斯风险最小化策略
  - 朴素贝叶斯的损失函数的第一项实际上是极大似然估计法，该方法用于进行参数估计，在该项前面加负号即可将该项的优化目标从最小变为最大，使用对数似然可以将连乘的优化目标变为连加形式，大大减小了求梯度的难度
- **区别:** 由于朴素贝叶斯是生成式模型，所以其拥有完全不同的正则项形式（由概率诱导）

其他方法损失函数

1. AdaBoost

$L = \sum_{i=1}^{n} \exp(-y_i f(\vec{x}_i))$

- 指数损失，强调误分类样本
- 自适应增加难样本权重
- AdaBoost 的指数损失使其更关注难以分类的样本，但这也可能导致过拟合噪声数据。
- AdaBoost 通常没有显式正则项，而是通过每一步更新权重的时候使用收缩因子控制每一步的学习步长，以此来控制模型的复杂度，这样做有很多好处：小步长更新避免对噪声样本的激进拟合；权重重分配更平滑，减少后续基学习器的震荡；可与任意基学习器（决策树/SVM 等）结合；无需修改损失函数或求解复杂优化问题

2. 聚类（K-means）

$L = \sum_{k=1}^{K} \sum_{\vec{x}_i \in C_k} \|\vec{x}_i - \vec{\mu}_k\|^2$

- 聚类的代价函数使用簇内平方距离之和，优化目标为尽可能使聚类变得更紧凑。由于聚类是无监督学习算法，故没有传统意义上的训练误差
- 聚类采取最小重构误差策略，使得聚类的分类更加准确

3. 网络机器学习

$\mathcal{Q}(F) = \frac{1}{2}\left(\sum_{i,j=1}^{n} W_{ij} \left\| \frac{1}{\sqrt{D_{ii}}} F_i - \frac{1}{\sqrt{D_{jj}}} F_j \right\|^2 + \mu \sum_{i=1}^{n} \|F_i - Y_i\|^2\right)$ 。

特点：

1. 结合图结构信息：损失函数的第一项利用图的权重矩阵 W 和度矩阵 D ，能够捕捉图结构中节点之间的关系，使得模型在学习过程中考虑到图的拓扑信息。

2. 平滑性约束：图拉普拉斯正则项对相邻节点的特征进行平滑约束，有助于在图结构上学习到连续和一致的特征表示。

3. 可调节的平衡：通过超参数 μ ，可以灵活地调整模型对图结构信息和数据拟合的重视程度。如果 μ 较大，模型更倾向于拟合训练数据；如果 μ 较小，则更注重图结构上的特征平滑。

4. 适用于图数据：这种损失函数特别适用于处理图数据，如图神经网络（GNNs）等模型中，在节点分类、图分类等任务中有着广泛的应用。

结论

- **损失函数的一般构成：**损失函数一般由训练损失和正则项两部分构成，训练损失部分为了使训练模型能很好处理数据集，而正则项负责惩罚训练参数，使得模型复杂度降低，防止模型对数据过拟合从而导致预测时表现不佳，一般损失函数还会引入一个超参数来平衡训练误差和正则项之间的比例关系，选取一个合适的超参数值会显著提升训练出的模型的效率
- **正则项的作用：**在训练时，训练数据集的某些数据噪音会被模型训练进去，用线性回归的模型举例，噪音可能使模型中训练的某个参数值异常增大，如果没有办法将使该参数值回归正常的话训练出的模型则会出现过拟合的现象（因为噪音也被模型很好地训练进去了，但我们不希望训练噪音），此时引入 L2 正则项对该异常增大的参数进行惩罚，在梯度下降的时候梯度中会含有与该参数有关的值，迭代过程中该参数就会自然而然地下降，由此 L2 正则项完成了对模型复杂度的降低，减少了过拟合的可能性
- **正则化:** SVM、LLSF、LR 和 NNet 均使用 L2 正则化（$\lambda\|\vec{\beta}\|^2$），但训练损失形式不同
- **NB 的独特性:** 概率诱导的正则项（$\|e^{\vec{\beta}}\|_1$）与参数范数完全不同
- **正则化影响:** 合理调整 λ 可显著提升性能（LLSF 调优后优于 SVM）