SPLBoost: An Improved Robust Boosting Algorithm Based on Self-paced Learning

Kaidong Wang, Yao Wang, Qian Zhao, Deyu Meng, Zongben Xu

CVPR 2017

Boosting

先从初始训练集训练出一个基学习器，再根据基学习器的表现对训练样本分布进行调整，使得先前基学习器做错的训练样本在后续受到更多关注，然后基于调整后的样本分布来训练下一个基学习器。最终将这T个基学习器进行加权结合

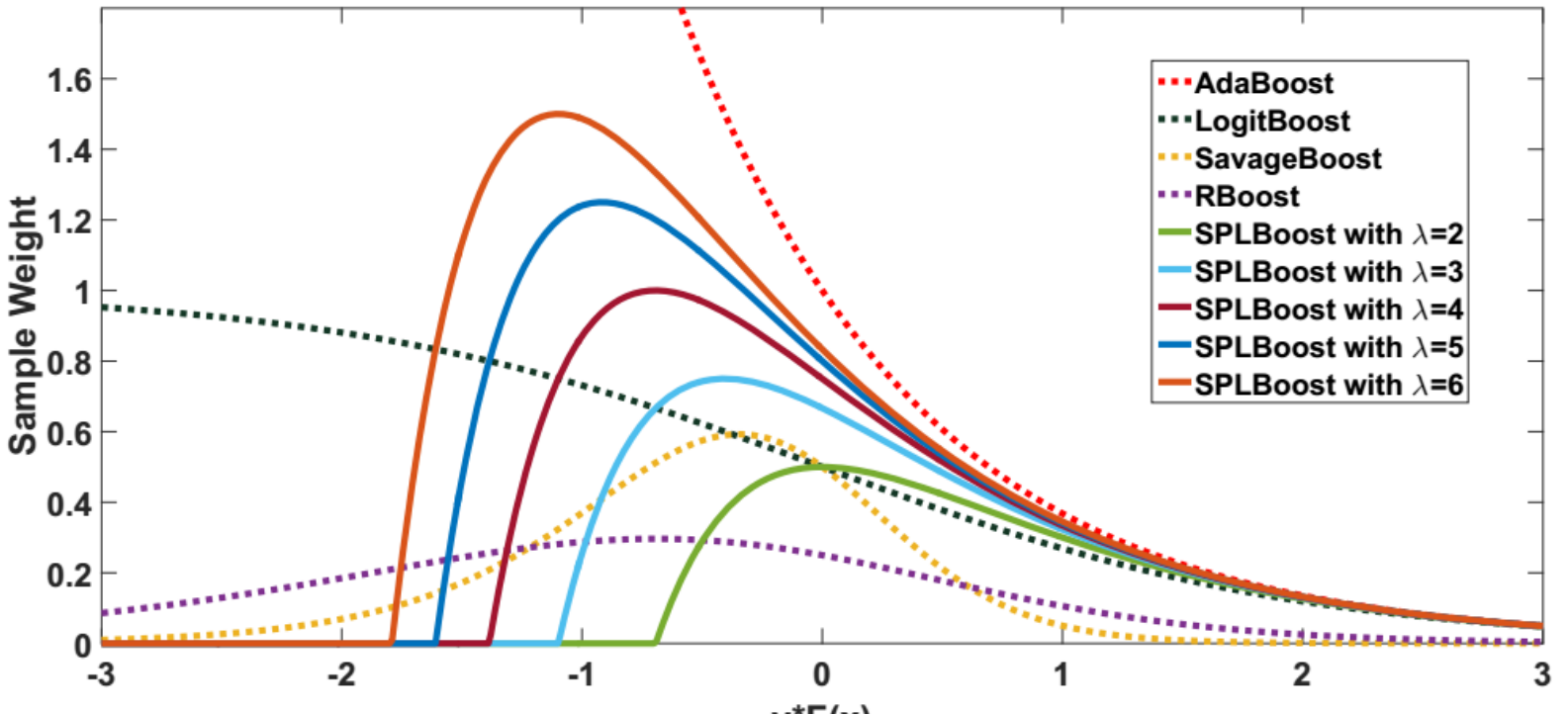
$$H(x) = \sum_{t=1}^{T} a_t h_t(x)$$

Boosting

各个鲁棒的不同Boosting算法的loss function

AdaBoost	$\varphi(yf(x)) = \exp(-yf(x))$
LogitBoost	$\varphi(yf(x)) = \log(1+\exp(yf(x)))$
SavageBoost	$\varphi(v) = \frac{1}{(1+e^{2v})^2}$
RBoost	$\varphi(v) = \frac{1}{(1+e^{v})^2}$

fewer penalties to those samples with negative margin



AdaBoost

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$
Initialize $D_1(i) = 1/m$.
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : X \to \{-1, +1\}$ with error

$$\epsilon_t = \Pr_{i \sim D_t} [h_t(x_i) \neq y_i].$$

- Choose $\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$.
- Update:

$$\begin{aligned} D_{t+1}(i) &= \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases} \\ &= \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t} \end{aligned}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

$$H(x) = \text{sign} \left(\sum_{t=1}^{T} \alpha_t h_t(x) \right).$$

SPLBoost

对Adaboost来说，新的弱分类器和对应的权要求最小化指数损失：

$$\{\alpha, f\} = \arg\min_{\alpha, f} \sum_{i=1}^{n} \mathrm{e}^{-y_i(F(x_i)+\alpha f(x_i))}.$$

SPLboost在原来Adaboost的基础上嵌入：

$$\{\alpha, f, \mathbf{v}\} = \arg\min_{\alpha, f} \sum_{i=1}^{n} v_i \mathrm{e}^{-y_i(F(x_i)+\alpha f(x_i))} + \hat{f}(v_i, \lambda)$$

$$v_i^* = \begin{cases} 1, & \mathrm{e}^{-y_i(F(x_i)+\alpha f(x_i))} < \lambda, \\ 0, & \text{otherwise.} \end{cases}$$

对于总是做错的样本，他们的v始终是0（消除对分类器的影响）使用轮流优化策略来优化

SPLBoost

$$\{\alpha, f, \mathbf{v}\} = \arg\min_{\alpha, f} \sum_{i=1}^{n} v_i \mathrm{e}^{-y_i(F(x_i)+\alpha f(x_i))} + \hat{f}(v_i, \lambda), \quad (4)$$

With fixed $\mathbf{v}$, (4) is a weighted exponential loss minimization problem:

$$\{\alpha, f\} = \arg\min_{\alpha, f} \sum_{i=1}^{n} v_i \mathrm{e}^{-y_i(F(x_i)+\alpha f(x_i))}. \quad (6)$$

SPLBoost

$$\{\alpha, f\} = \arg\min_{\alpha, f} \sum_{i=1}^{n} v_i \mathrm{e}^{-y_i(F(x_i)+\alpha f(x_i))}. \qquad (6)$$

对固定的a，将上式按泰勒公式展开至二阶：

$$f = \arg\min_{f} \sum_{i=1}^{n} v_i \mathrm{e}^{-y_i(F(x_i)+\alpha f(x_i))}$$

$$\approx \arg\min_{f} \sum_{i=1}^{n} v_i \mathrm{e}^{-y_i(F(x_i))}(1 - y_i\alpha f(x_i) + \alpha^2 f(x_i)^2/2). \qquad (7)$$

$v_i e^{-y_i(F(x_i)+\alpha f(x_i))} = v_i e^{-y_i F(x_i)} \cdot e^{-y_i \alpha f(x_i)}$ 这一项展开

因为e^x的级数展开，能够成立的区间是$(-\infty, +\infty)$:

$$e^x = 1 + x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \frac{1}{4!}x^4 + \frac{1}{5!}x^5 + \frac{1}{6!}x^6 + \cdots\cdots$$

SPLBoost

$$\approx \arg\min_f \sum_{i=1}^{n} v_i e^{-y_i(F(x_i))}(1 - y_i \alpha f(x_i) + \alpha^2 f(x_i)^2/2).$$

Taking $f(x) \in \{-1, 1\}$ into consideration, we have

$$\begin{aligned} f &= \arg\min_f \sum_{i=1}^{n} v_i e^{-y_i(F(x_i))}(y_i - \alpha f(x_i))^2 \\ &= \arg\min_f \sum_{i=1}^{n} v_i e^{-y_i(F(x_i))}(y_i - f(x_i))^2 \qquad (8) \\ &= \arg\min_f \sum_{i=1}^{n} v_i w_i (y_i - f(x_i))^2, \end{aligned}$$

where $w_i = e^{-y_i(F(x_i))}$ which is the same as the sample weights defined in AdaBoost. We can find that solving the

$$y_i^2 = 1$$

$$\left(1 - y_i \alpha f(x_i) + \frac{\alpha^2 f(x_i)^2}{2}\right) = \left[\left(y_i - \frac{\alpha}{2} f(x_i)\right)^2 + \frac{\alpha^2 f(x_i)^2}{4}\right]$$

SPLBoost

Solve weighted least squares problem to obtain the weak learner f(x) is equivalent to implement AdaBoost except for the use of latent weight variable v. Thus, imitating the approach in AdaBoost, one can train f(x) from the training data with sample weights viwi rather than wi. Given f(x) ∈ {-1,1}, we can directly optimize (6) to determine α:

$$\{\alpha, f\} = \arg\min_{\alpha, f} \sum_{i=1}^{n} v_i \mathrm{e}^{-y_i(F(x_i)+\alpha f(x_i))}. \tag{6}$$

$$w_i = \mathrm{e}^{-y_i(F(x_i))}$$

$$\begin{aligned} \alpha &= \arg\min_{\alpha} \sum_{i=1}^{n} v_i w_i \mathrm{e}^{-y_i \alpha f(x_i)} \\ &= \arg\min_{\alpha} \sum_{y_i = f(x_i)} v_i w_i \mathrm{e}^{-\alpha} + \sum_{y_i \neq f(x_i)} v_i w_i \mathrm{e}^{\alpha}. \end{aligned} \tag{9}$$

It is not hard to see the above objective is a convex function, thus to get the optimal α, we can directly calculate its derivative and set it to be zero:

$$-\sum_{y_i = f(x_i)} v_i w_i \mathrm{e}^{-\alpha} + \sum_{y_i \neq f(x_i)} v_i w_i \mathrm{e}^{\alpha} = 0. \tag{10}$$

SPLBoost

$$-\sum_{y_i=f(x_i)} v_i w_i e^{-\alpha} + \sum_{y_i\neq f(x_i)} v_i w_i e^{\alpha} = 0. \tag{10}$$

$$\begin{aligned}\alpha &= \frac{1}{2}\log\frac{\sum_{y_i=f(x_i)} v_i w_i}{\sum_{y_i\neq f(x_i)} v_i w_i} \\ &= \frac{1}{2}\log\frac{1-\sum_{y_i\neq f(x_i)} v_i w_i}{\sum_{y_i\neq f(x_i)} v_i w_i} \\ &= \frac{1}{2}\log\frac{1-\mathrm{err}}{\mathrm{err}},\end{aligned} \tag{11}$$

where $\mathrm{err} = \sum_{y_i\neq f(x_i)} v_i w_i$ is the weighted misclassification error of weak learner $f(x)$. It is easy to see that the formula

SPLBoost

$$F(x) \leftarrow F(x) + \alpha f(x). \tag{12}$$

In the next outer iteration, the latent weight variable $\mathbf{v}$ can be initialized to the current $\mathbf{v}$ and w_i is updated as follows:

$$w_i \leftarrow w_i \mathrm{e}^{-\alpha y_i f(x_i)}. \tag{13}$$

Since $-y_i f(x_i) = 2 \times 1_{y_i \neq f(x_i)} - 1$, the update is equivalent to

$$w_i \leftarrow w_i \exp\left(\log \frac{1-\mathrm{err}}{\mathrm{err}} 1_{y_i \neq f(x_i)}\right). \tag{14}$$

This update of w is clearly the same as that in AdaBoost.

SPLBoost

Algorithm 1 SPLBoost algorithm

Input: training samples $\{(x_1, y_1), \cdots, (x_n, y_n)\}$, iteration count T, parameter λ;

Initialization: $w_i = 1/n$, $v_i = 1$, $i = 1, \cdots, n$;

for t=1 to T **do**

while not converge **do**

1. Fit the classifier $f_t(x) \in \{-1, 1\}$ using weights $v_i w_i / (\sum_i v_i w_i)$ on the training data;
2. Compute err $= \sum_{y_i \neq f(x_i)} v_i w_i / \sum_i v_i w_i$, $\alpha_t = \frac{1}{2} \log \frac{1-\text{err}}{\text{err}}$;
3. Compute $\mathbf{v}$;

end while

4. Set $w_i \leftarrow w_i \exp\left(\log \frac{1-\text{err}}{\text{err}} 1_{y_i \neq f(x_i)}\right)$;

end for

Output: The strong classifier $sign \sum_{t=1}^{T} \alpha_t f_t(x)$;

Experiment

1. 高斯数据(toy data)

According to the 2-D Gaussian distributions

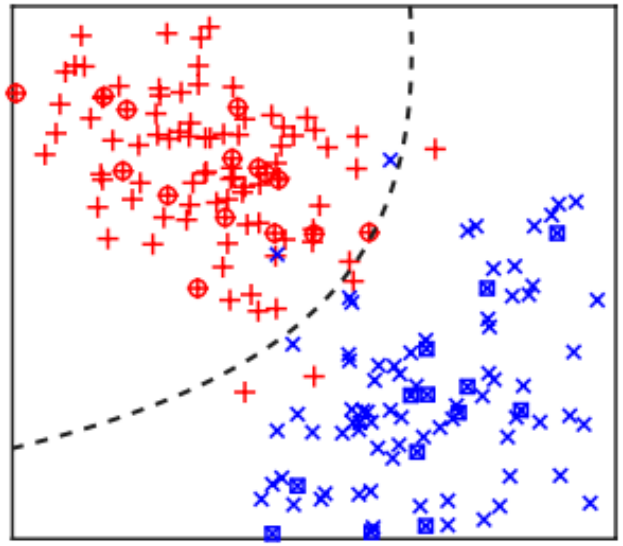
$$N\left([2,-2],\begin{bmatrix}2.5 & 1.5\\ 1.5 & 5\end{bmatrix}\right)$$

and

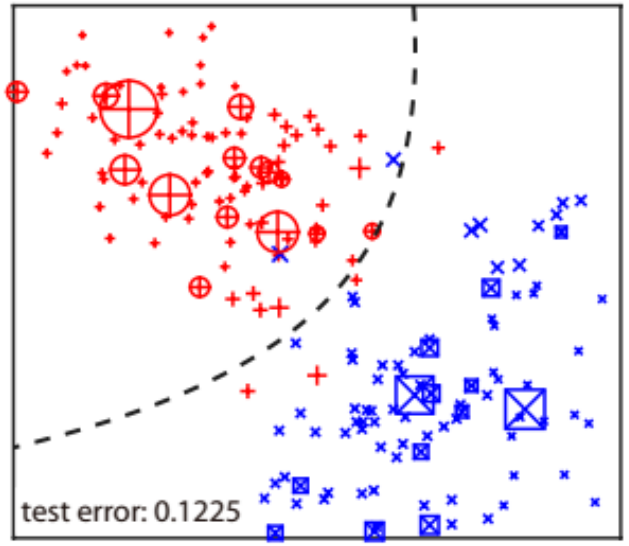
$$N\left([-2,2],\begin{bmatrix}2.3 & -0.7\\ -0.7 & 2.3\end{bmatrix}\right),$$

we first generate 100 samples for both the negative and positive classes, then randomly select 15% from both the two classes and reverse their labels, and those samples selected can be considered to be outliers. In this way we have obtained the

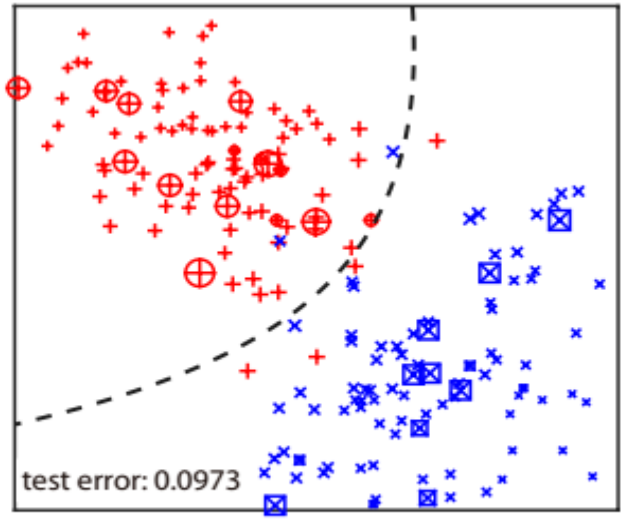
Experiment



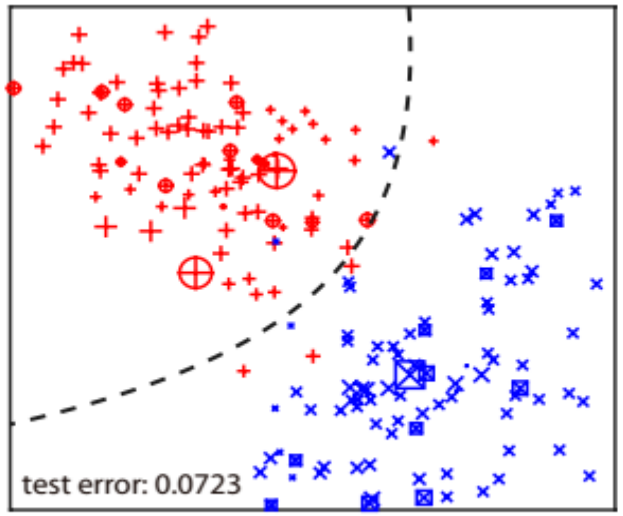
(a) original data



(b) AdaBoost



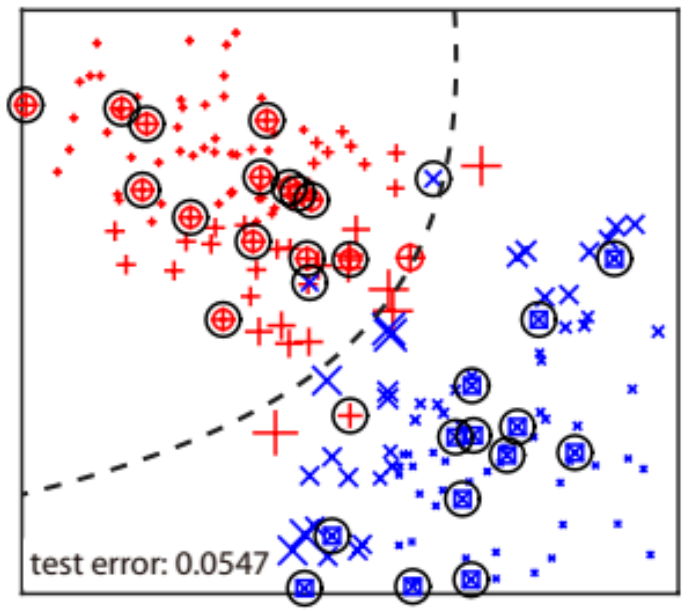
(c) LogitBoost



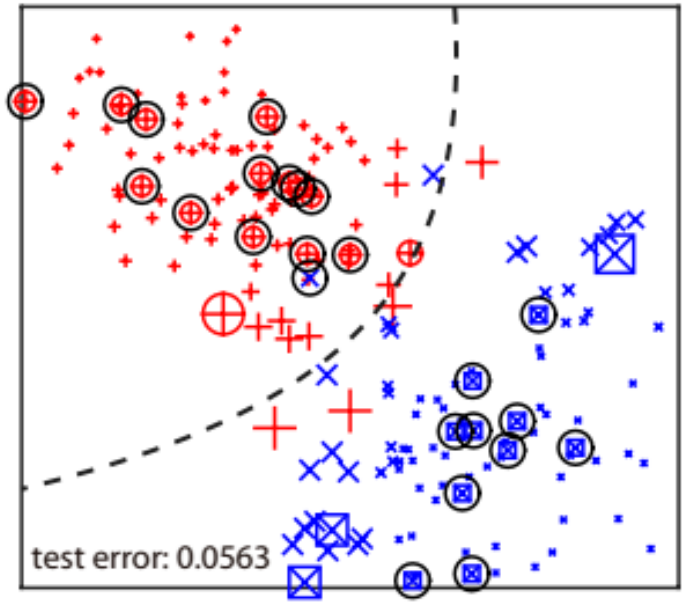
(d) SavageBoost

- positive samples
- negative samples
- negative samples(outlier)
- positive samples(outlier)
- samples with 0 weights in SPLBoost
- -Bayes decision boundary

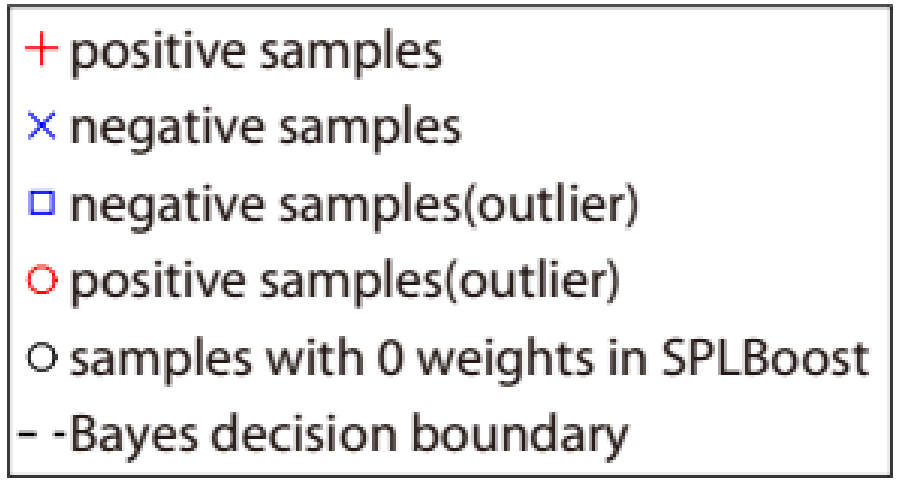
Experiment



(g) SPLBoost with $\lambda = 1.5$



(i) SPLBoost with $\lambda = 2.5$



Lamda小时，算法更加严格，更多的样本会被消除

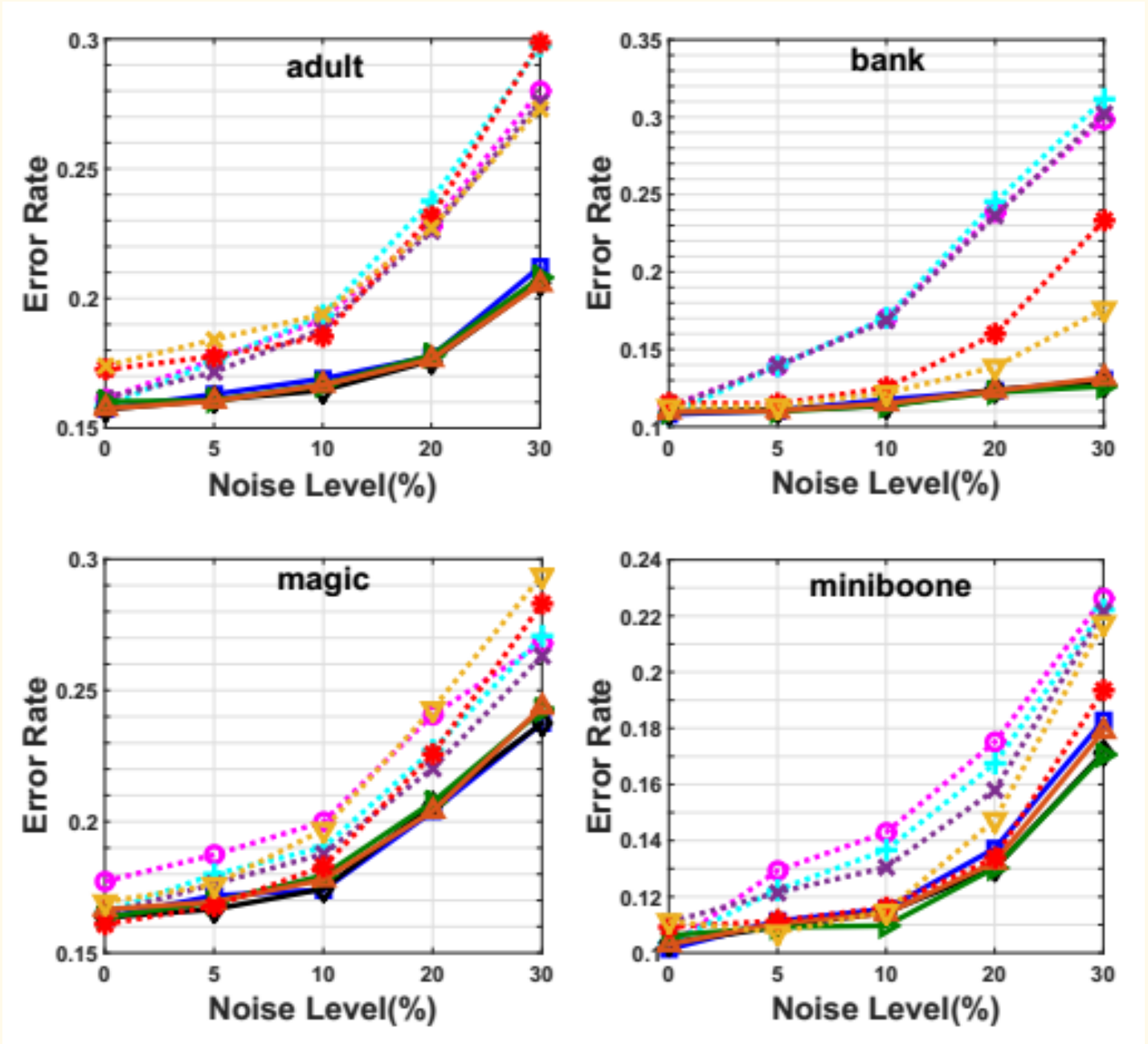
Experiment

2.UCI数据集

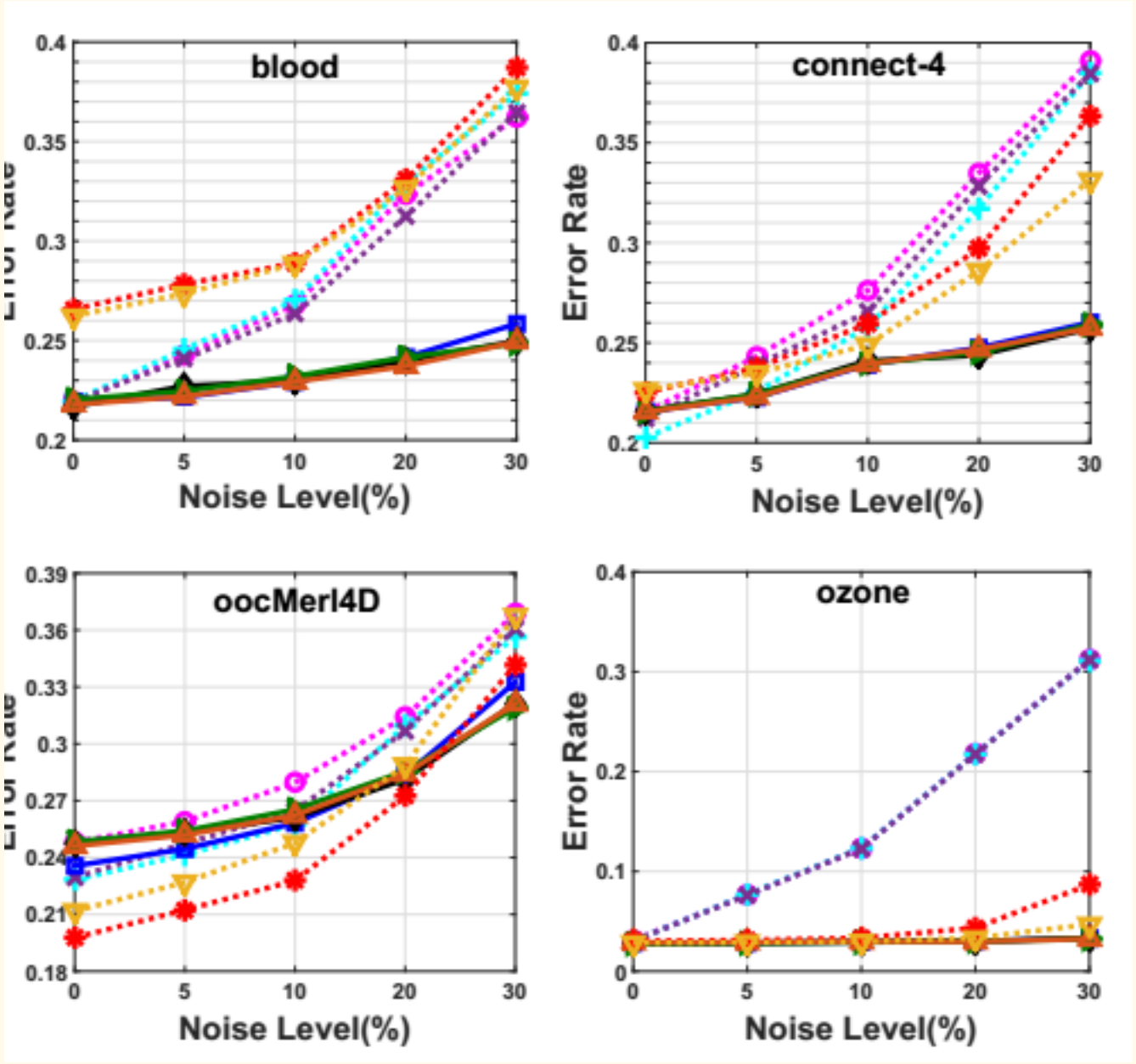
17 UCI datasets, train/test = 7/3 , noise level=[0,5,10,20,30] ,
use 5-fold crossvalidation to choose parameters.

the weak learner of AdaBoost, SPLBoost and RobustBoost is chosen as C4.5, while for LogitBoost and RBoost, the regression tree CART is used as the weak learner.

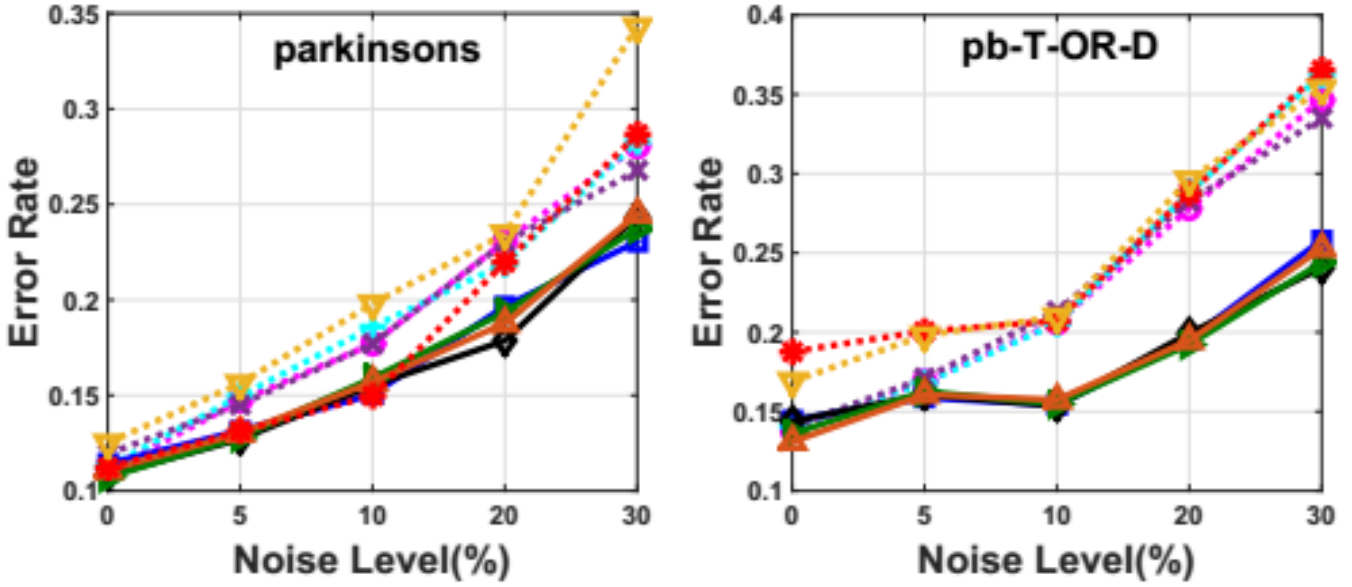
Experiment


adult
bank
magic
miniboone
Error Rate
Noise Level(%)
AdaBoost
LogitBoost
RobustBoost
SavageBoost
RBoost
SPLBoost-hard
SPLBoost-linear
SPLBoost-polynomial with t=1.3
SPLBoost-polynomial with t=4

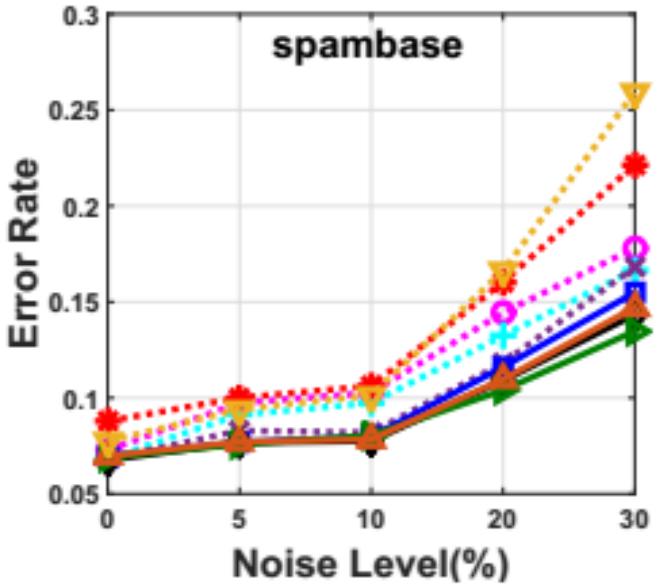
Experiment

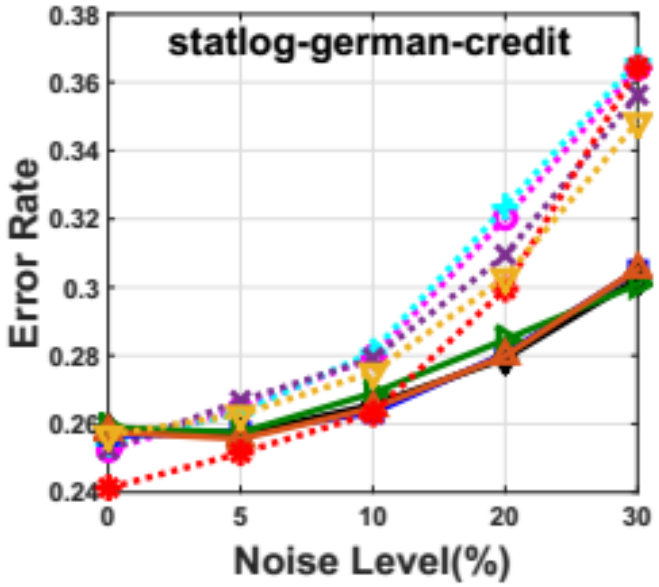

blood
connect-4
oocMerl4D
ozone
Error Rate
Noise Level(%)
AdaBoost
LogitBoost
RobustBoost
SavageBoost
RBoost
SPLBoost-hard
SPLBoost-linear
SPLBoost-polynomial with t=1.3
SPLBoost-polynomial with t=4

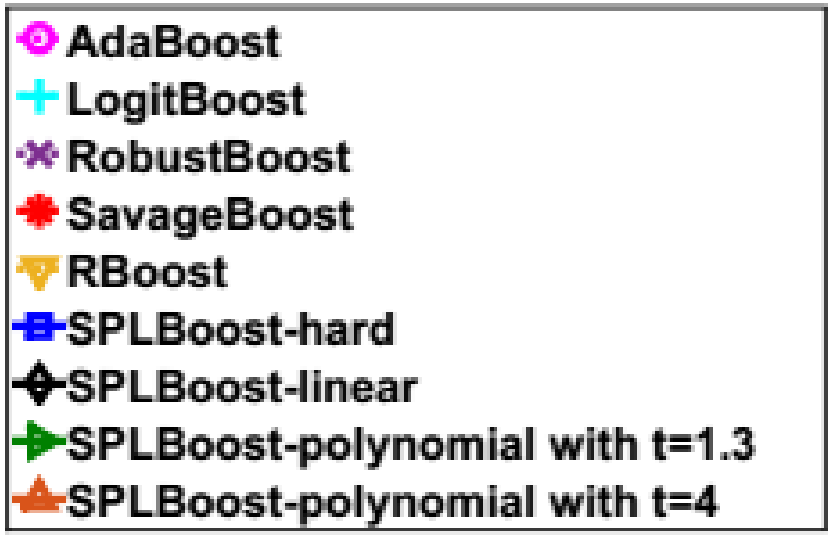
Experiment


parkinsons
Error Rate
Noise Level(%)
0.35
0.3
0.25
0.2
0.15
0.1
0
5
10
20
30

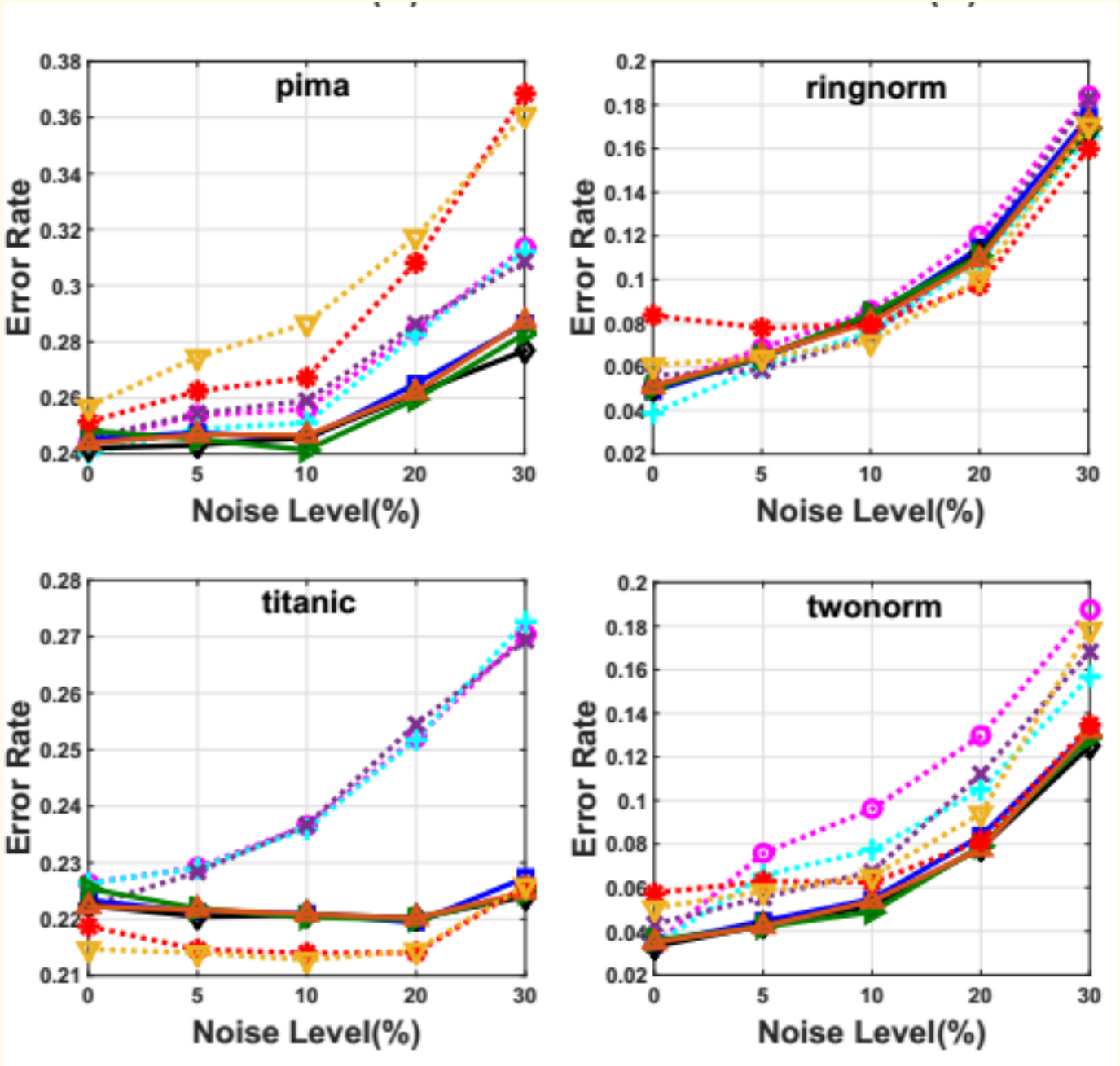
pb-T-OR-D
Error Rate
Noise Level(%)
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0
5
10
20
30

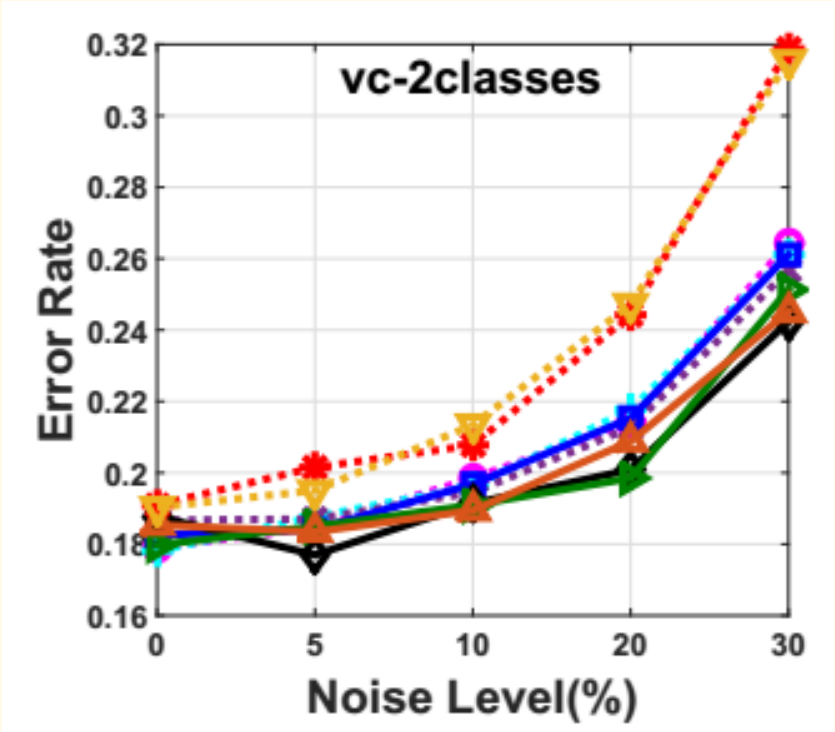

spambase
Error Rate
Noise Level(%)
0.3
0.25
0.2
0.15
0.1
0.05
0
5
10
20
30

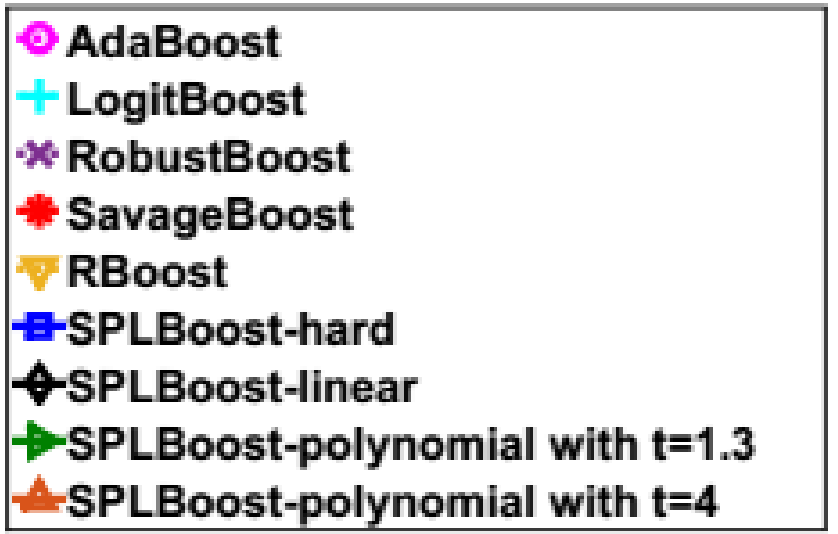

statlog-german-credit
Error Rate
Noise Level(%)
0.38
0.36
0.34
0.32
0.3
0.28
0.26
0.24
0
5
10
20
30


AdaBoost
LogitBoost
RobustBoost
SavageBoost
RBoost
SPLBoost-hard
SPLBoost-linear
SPLBoost-polynomial with t=1.3
SPLBoost-polynomial with t=4

Experiment


pima
ringnorm
titanic
twonorm
Error Rate
Noise Level(%)


vc-2classes
Error Rate
Noise Level(%)


AdaBoost
LogitBoost
RobustBoost
SavageBoost
RBoost
SPLBoost-hard
SPLBoost-linear
SPLBoost-polynomial with t=1.3
SPLBoost-polynomial with t=4

Conclusion

AdaBoost具有很强的学习性能，但易受到噪声数据的影响，SPL通过样本选择的方式帮助AdaBoost减轻噪声的影响，从而具有更好的鲁棒性，在强噪声的情况下性能不会下降太多。
但在无噪声的情况下，这样做并不能带来性能提升，甚至会下降

Thank you