Ensemble Methods

Harish Guruprasad
IIT Madras

Machine Learning in Practice

- Real world machine learning problems rarely have a unique and single best solution.

- Multiple thought processes and teams and approaches often yield equally valid but completely different solutions.

- The set of methods for combining many such solutions (classifiers, regressors etc.) into one solution are known as “Ensemble methods”

The Netflix Challenge

- Data set of ~100M star ratings that ~500K users gave to ~18K movies.

- Training data: <*user, movie, date of grade, grade*>

- The grand prize of $1,000,000, to be given to a team which beat Netflix's rating prediction algorithm by 10%

Netflix challenge: Winning Solution

- The winning team — “BellKor’s Pragmatic Chaos” (itself a merger of several teams) combined a total of **107** separate prediction models!!

- The methods used various approaches — factor models, regression models, neighbourhood models, etc.

- Most other teams solutions also included large numbers of disparate models combined together.

- Ensemble methods are almost always the state of the art in any large scale Machine learning problem.

Paradigms of Ensemble Methods

- **Parallel Ensembles**
  - Combines several models, each of which were built separately to fit the data.
  - Allows Naive parallelisation.
  - Example: Bagging.
- **Sequential Ensembles**
  - Each model built in the ensemble explicitly uses the other models and tries to make up for the weaknesses of the rest of the models.
  - Example: Boosting

Paradigms of Ensemble Methods

- **Parallel Ensembles**
  - Combines several models, each of which were built separately to fit the data.
  - Allows Naive parallelisation.
  - Example: Bagging.
- **Sequential Ensembles**
  - Each model built in the ensemble explicitly uses the other models and tries to make up for the weaknesses of the rest of the models.
  - Example: Boosting

Parallel Ensembles : Bagging

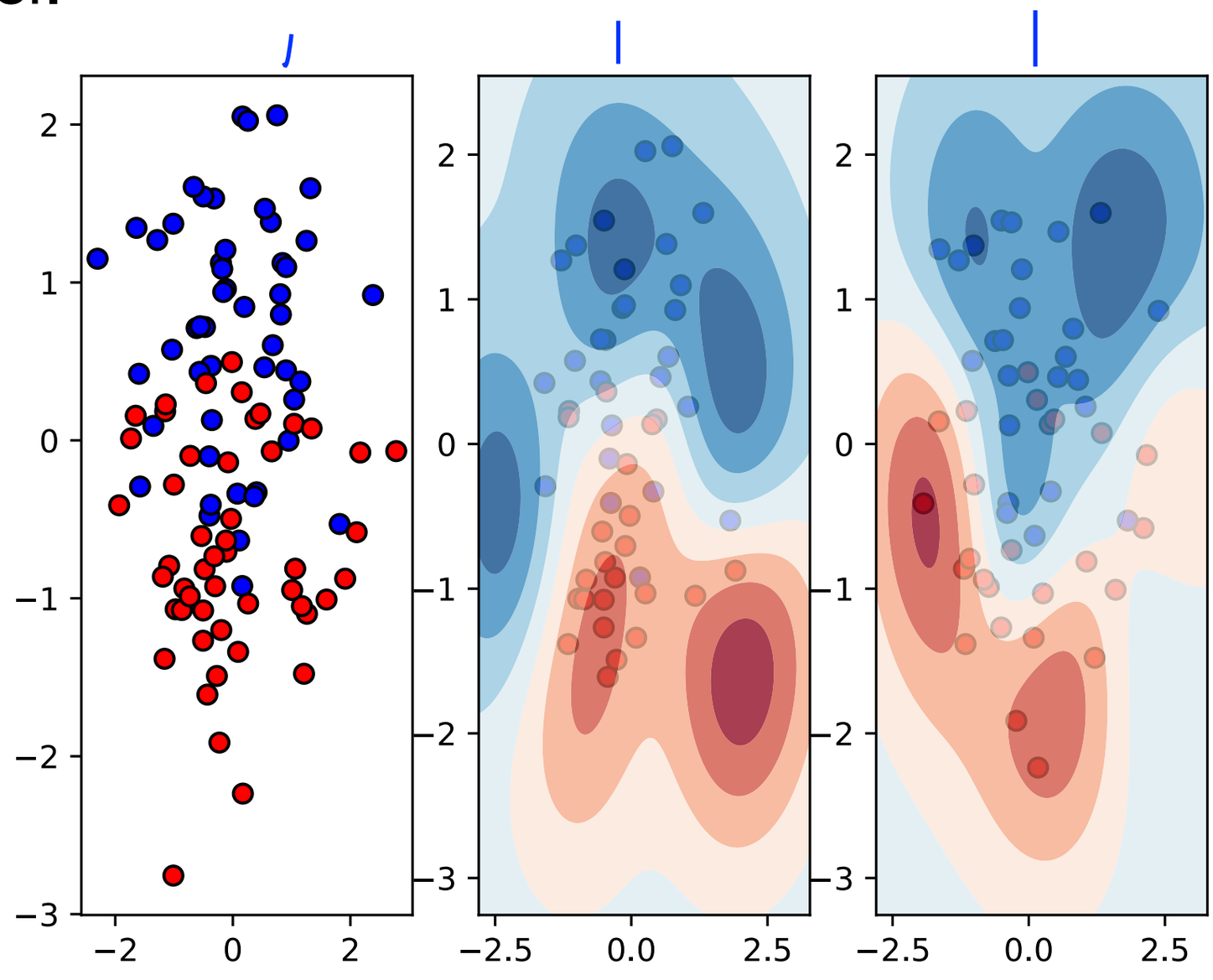
- Learners are often “high-capacity” and generally overfit.
- One option is to regularise the learner somehow.
  - Not always clear how to do so for several learners.
  - May not be the best approach.

- Bagging:
  - Combine several separate models learned on some subset of the data simply by averaging (or taking a majority vote).
  - Key insight: High capacity learners give very distinct models when trained on different subsets of the training data.

Unstable Learners

- Gaussian data centred at (1,0) and (-1,0).

- Subsets of data are used for learning by a “high-capacity” classifier.



Bagging with Bootstrap Sampling

(Use as rough space)

Bagging with Bootstrap Sampling

$$S = \{(x_1, y_1), \ldots, (x_m, y_m)\} \sim D^m$$

$S \approx$

b_S: Randomly chosen subset of S

$$H(x) = \mathbb{E}_{b_S}[h_{b_S}(x)]$$

$$\begin{aligned}
(\eta(x) - H(x))^2 &= \left(\mathbb{E}_{b_S}[\eta(x) - h_{b_S}(x)]\right)^2 \\
&\leq \mathbb{E}_{b_S}\left[(\eta(x) - h_{b_S}(x))^2\right] \\
\mathrm{MSE}[H] &\leq \mathbb{E}_{b_S}\left[\mathrm{MSE}[h_{b_S}]\right]
\end{aligned}$$

MSE of bagged model is no worse than the average MSE of individual models.

When Does Bagging Work?

$$\mathrm{MSE}[H] << \mathbb{E}_{b_S}\left[\mathrm{MSE}[h_{b_S}]\right]$$

$$\Updownarrow$$

$$\left(\mathbb{E}_{b_S}[\eta(x) - h_{b_S}(x)]\right)^2 << \mathbb{E}_{b_S}\left[(\eta(x) - h_{b_S}(x))^2\right]$$

$$\Updownarrow$$

$$\mathrm{var}_{b_S}[h_{b_S}(x)] >> 0$$

Bagged Decision Trees

- Decision trees are a good candidate for bagging:
  - Very easy to overfit, and difficult to regularise.
  - Highly unstable learner.
- Bagged decision trees by themselves are powerful.
  - But…

Random Feature Selection

- To increase the variance, each individual node of each tree is learned on only a subset of features.

- This helps in case there is a highly predictive feature, as all learners will use this feature and hence will be correlated.

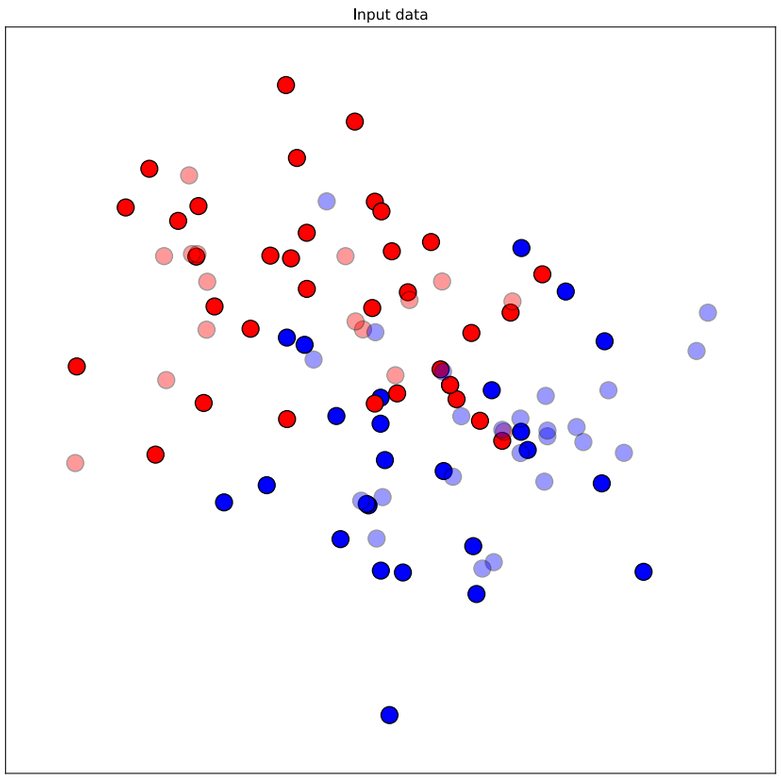
- The resulting ensemble method is called the Random forest regressor/classifier.

Random Forest Regressor/Classifier

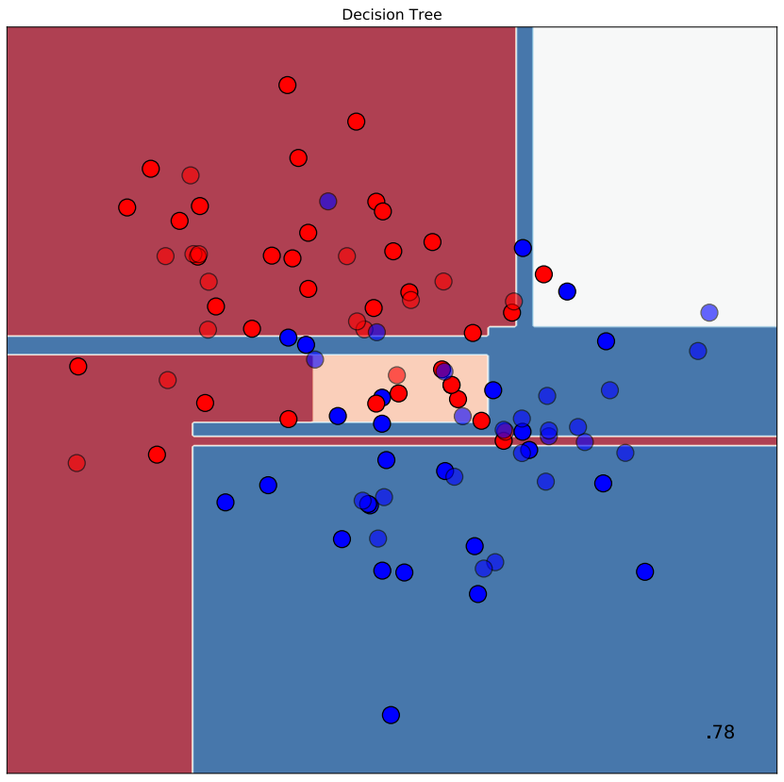
m= #datpoints,
d= #dimensions
Typical: a,b=0.5

- Repeat “n_est” times:
    - Select “a*m” samples at random (with replacement, preferably)
    - Learn a decision tree with the above sample, with a small variant:
        - while constructing each node randomly select “b*d” attributes and choose the best split only among these.
- Make prediction by averaging the “n_est” learned models.

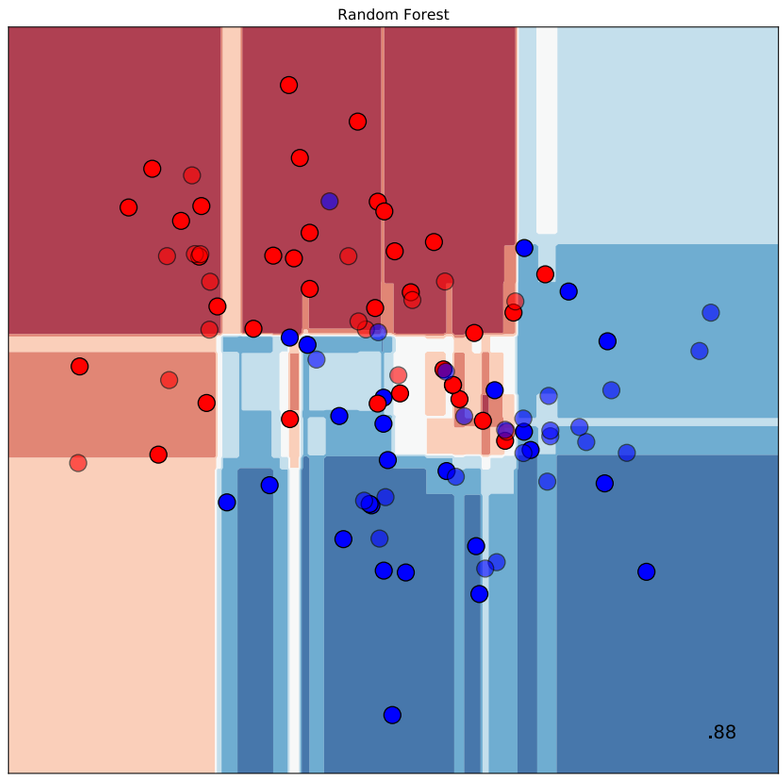
Random Forest Classifier Example



Decision tree



Random forest



Accuracy: **78%** **88%**

Paradigms of Ensemble Methods

- **Parallel Ensembles**
  - Combines several models, each of which were built separately to fit the data.
  - Allows Naive parallelisation.
  - Example: Bagging.

- **Sequential Ensembles**
  - Each model built in the ensemble explicitly uses the other models and tries to make up for the weaknesses of the rest of the models.
  - Example: Boosting

Boosting

- Assume you have access to a learner, that gives a classifier fitting training data.
    - But it is “weak”, i.e. it can only do slightly better than random (50%).
- Is there a way to create multiple such “weak” classifiers and “boost” them to a stronger one?
- A parallel/oblivious strategy will not be successful here.
- Each new weak classifier built must strategically cover for other classifiers weaknesses.

Boost by Majority

- Key idea:
  - The weak learners are given weighted training samples.
  - Weight of a sample depends on how previous weak learners have done on that sample.

Boost by Majority

Input:

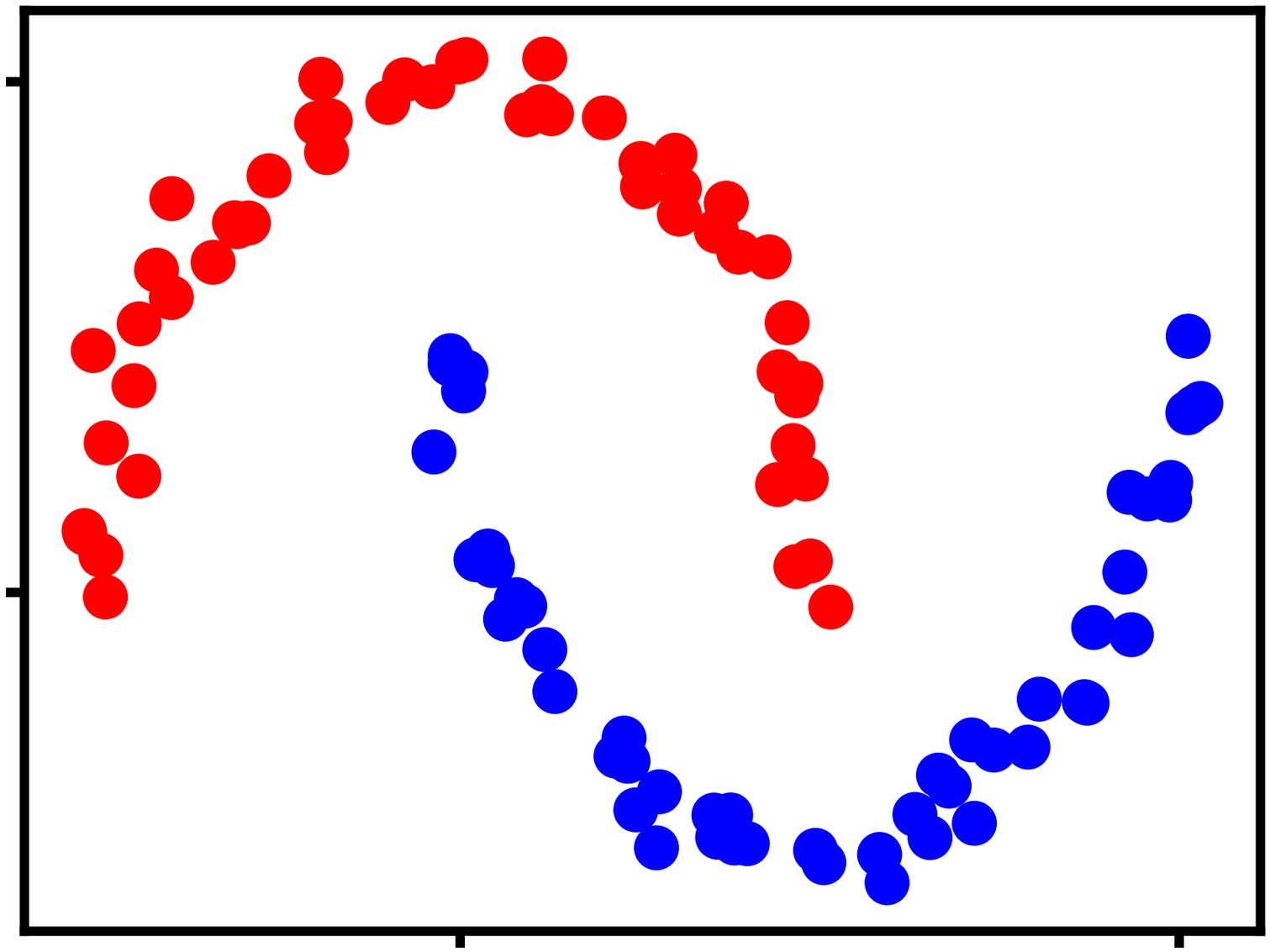
Weak learner with accuracy greater than $0.5 + \gamma$.
Training sample $S = (x_1, y_1), \ldots, (x_m, y_m) \in \mathcal{X} \times \{-1, +1\}$

$\beta = \frac{0.5+\gamma}{0.5-\gamma}$, $h_t : \mathcal{X} \to \{-1, +1\}$

Algorithm:

$\mathbf{w}^1 = \mathbf{1}$
For $t = 1$ to T:
$h_t = \text{WeakLearner}(S, \mathbf{w}^t)$
$l_{t,i} = |h_t(x_i) - y_i|/2$
$w_i^{t+1} = w_i^t \beta^{l_{t,i}}$
End For
Output: $h(x) = \text{sign}\left(\sum_{t=1}^{T} (h_t(x))\right)$

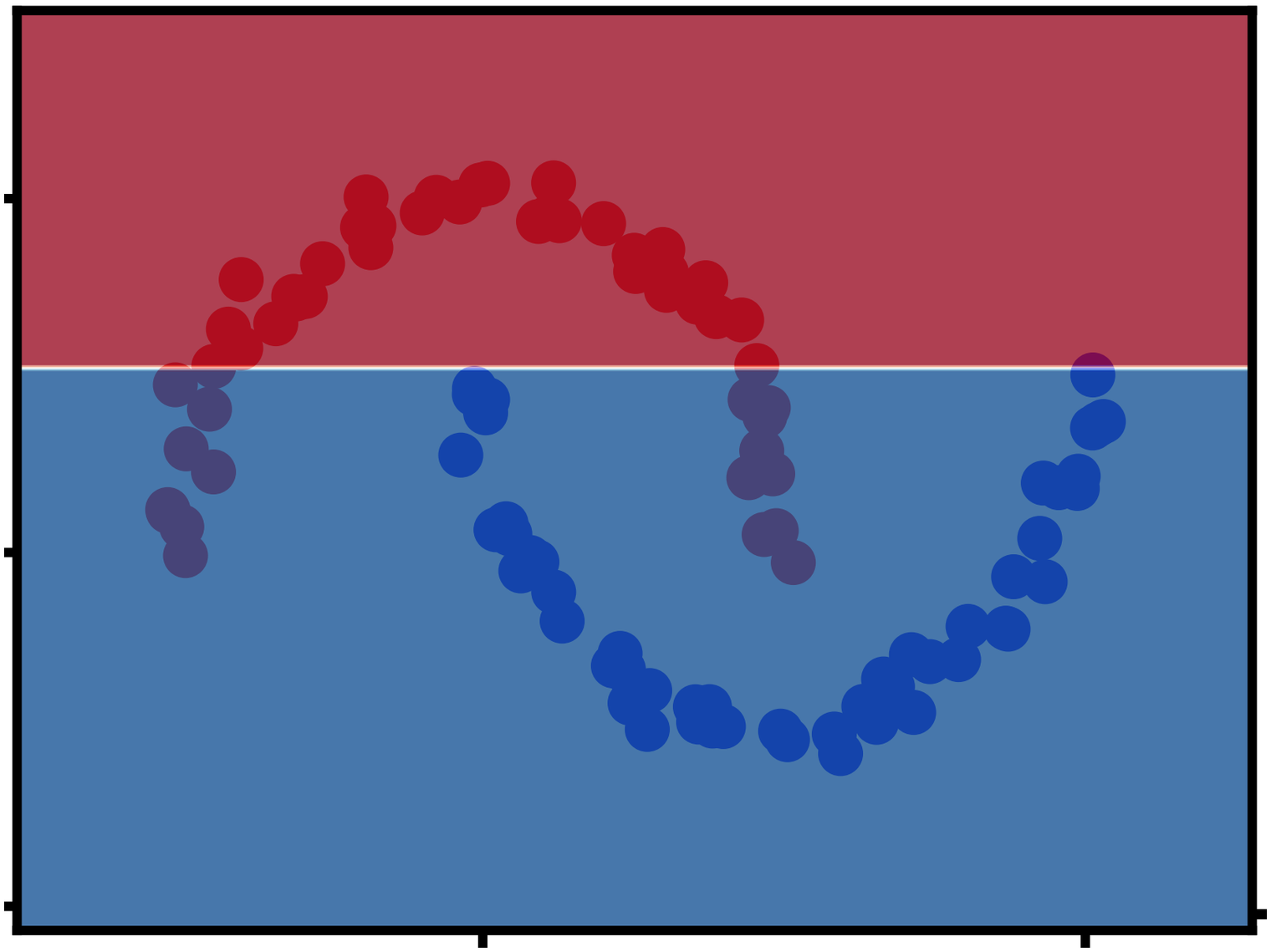
Example



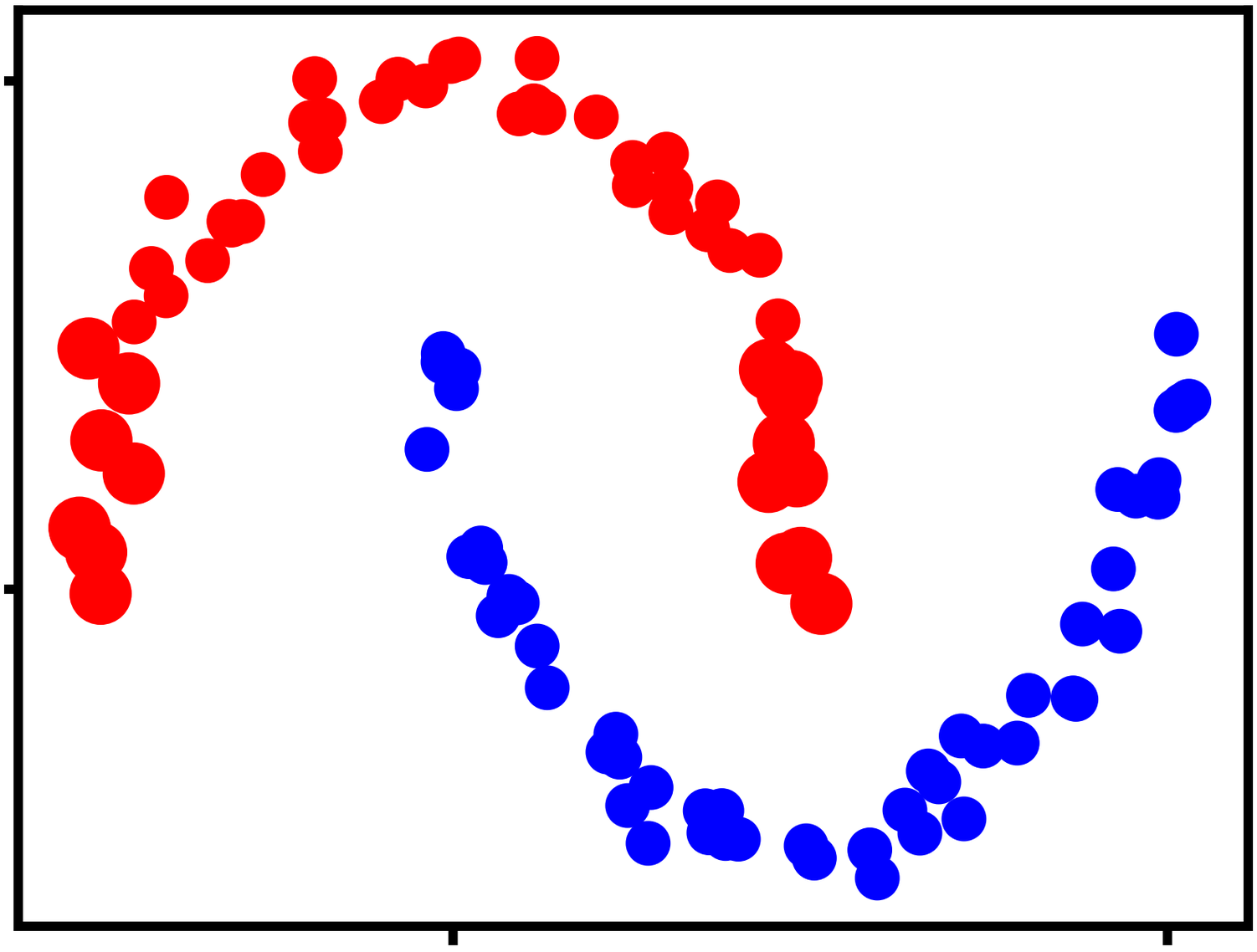
Weak learner: Decision stumps

$$\gamma = 0.2, \beta = 2.33$$

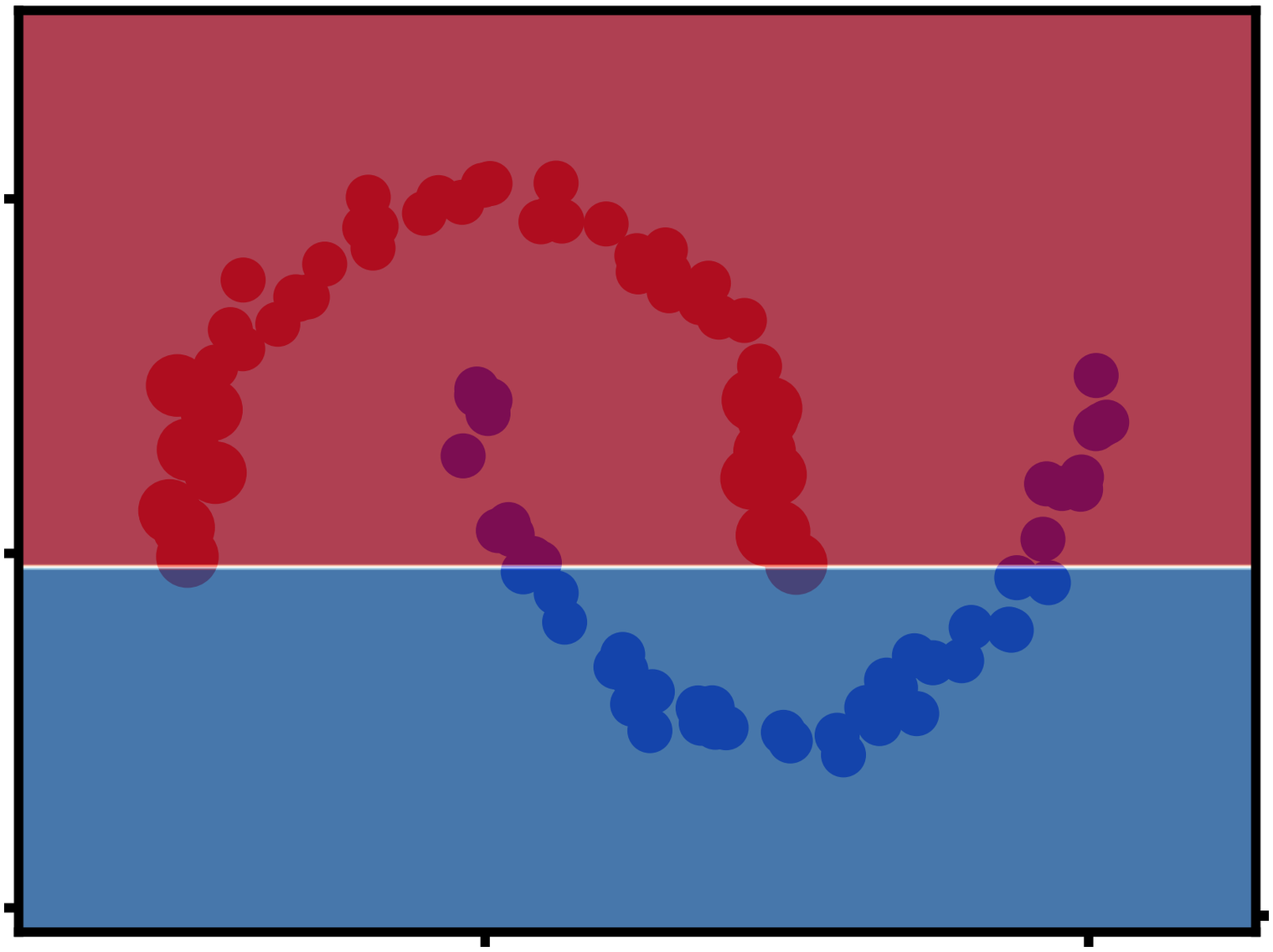
Weak classifier 1



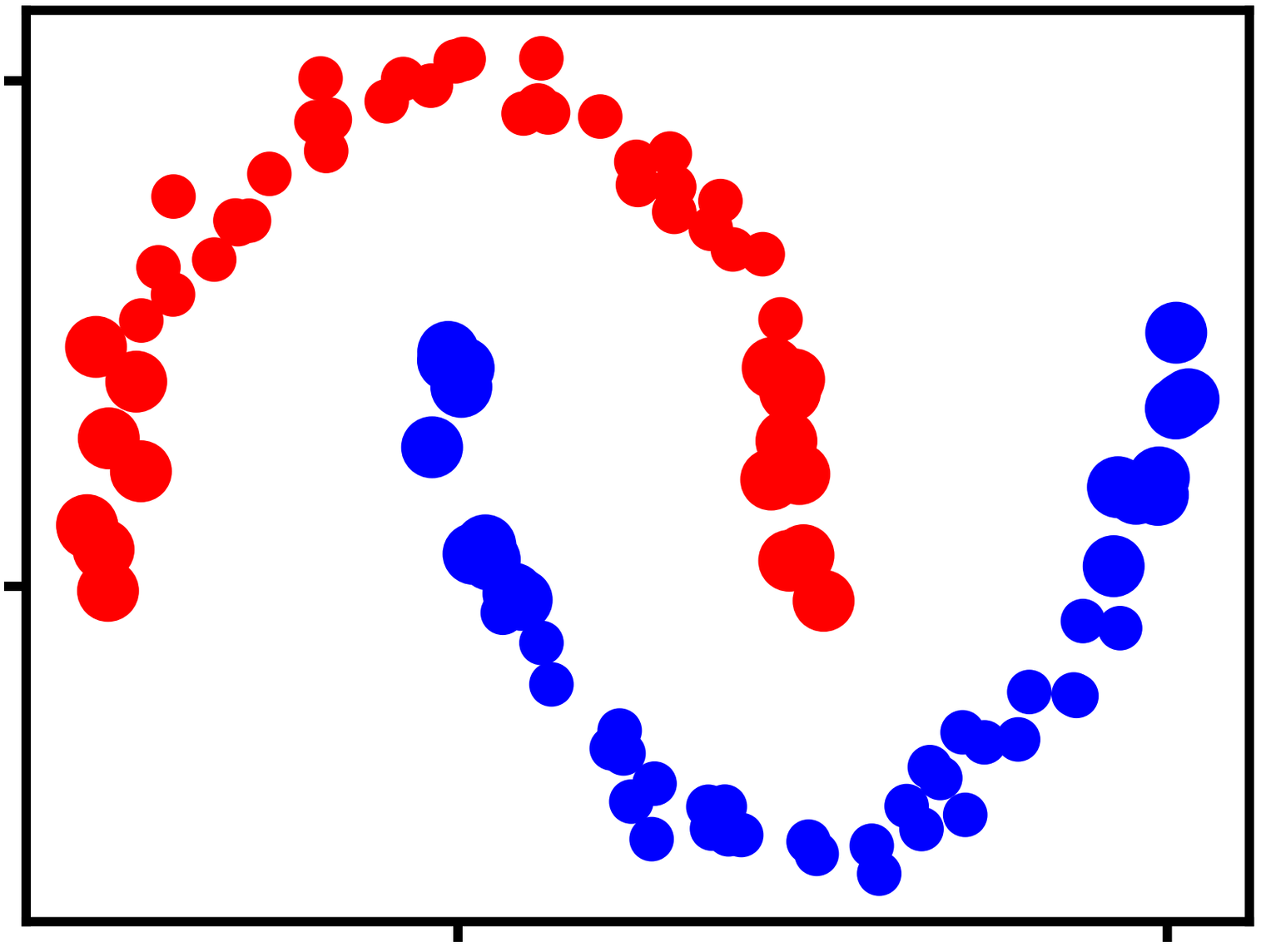
Weighted data 2



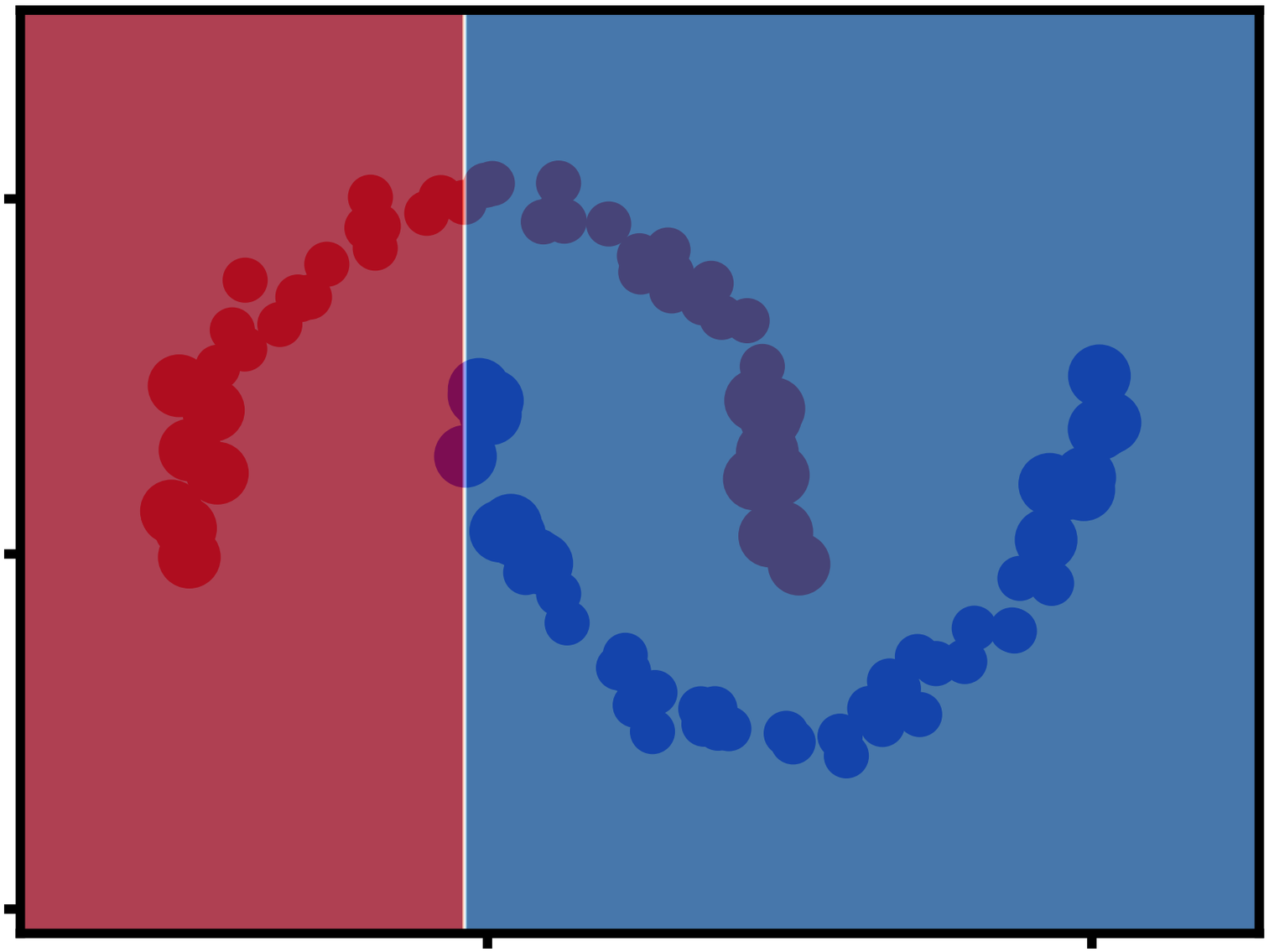
Weak classifier 2



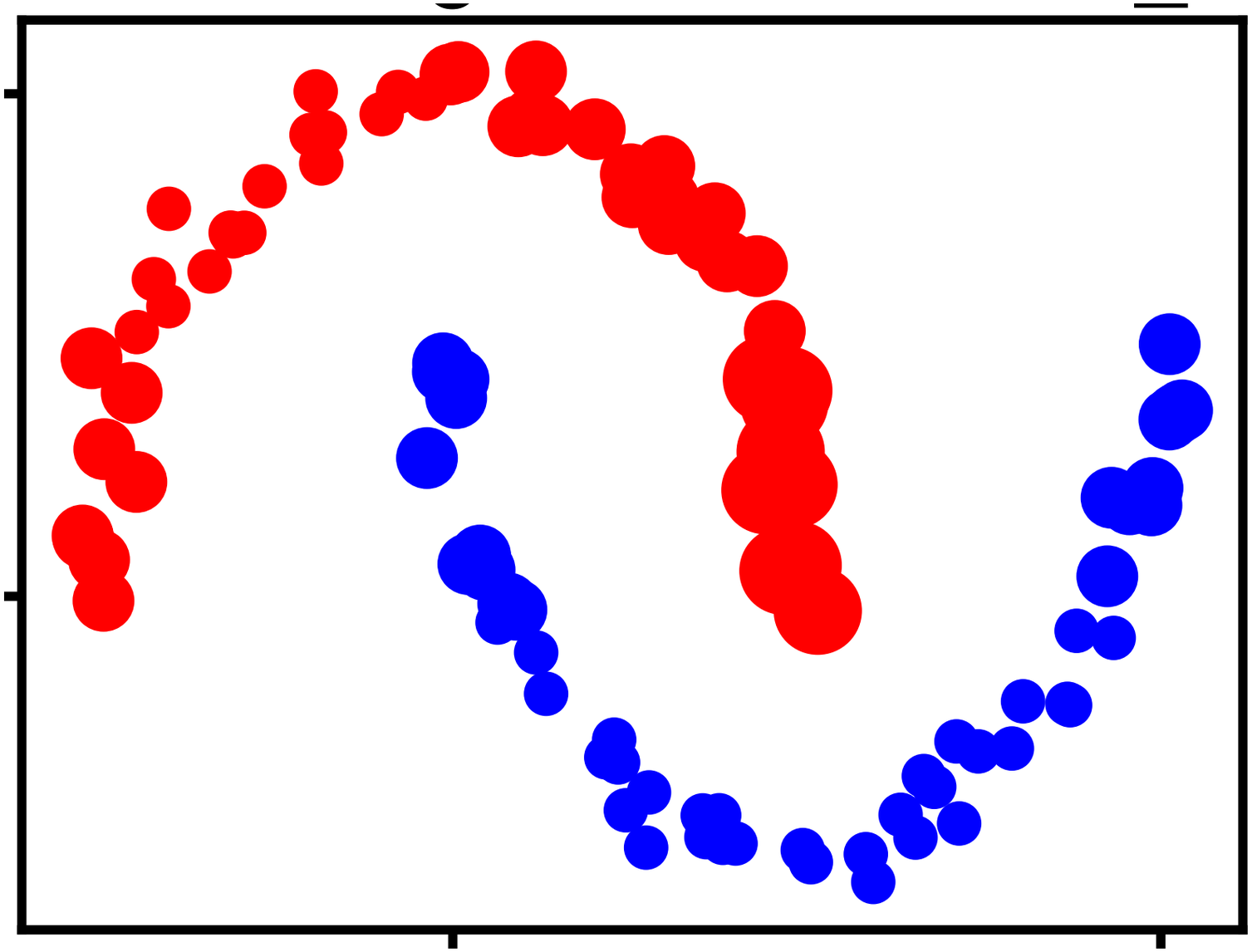
Weighted data 3



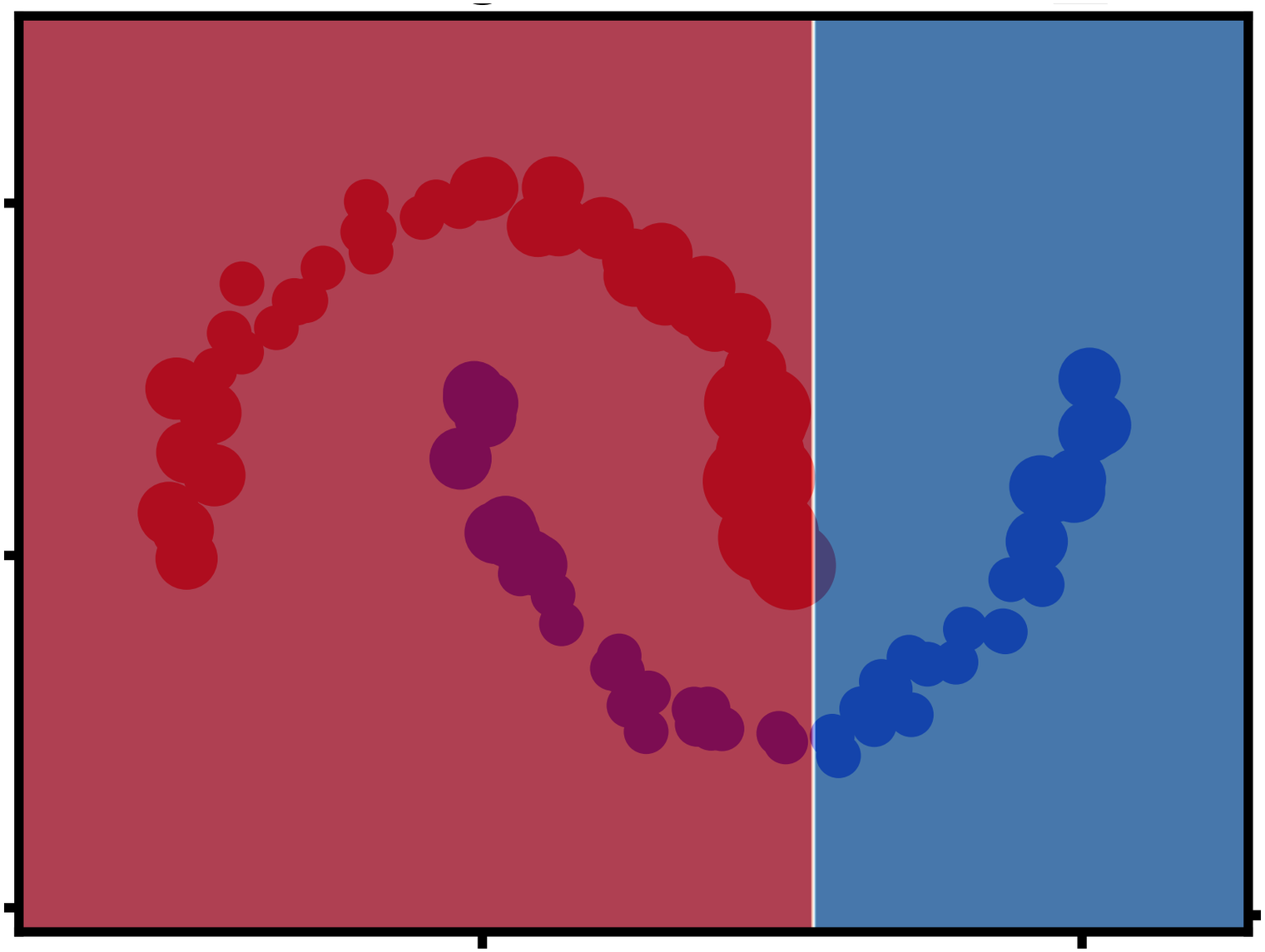
Weak classifier 3



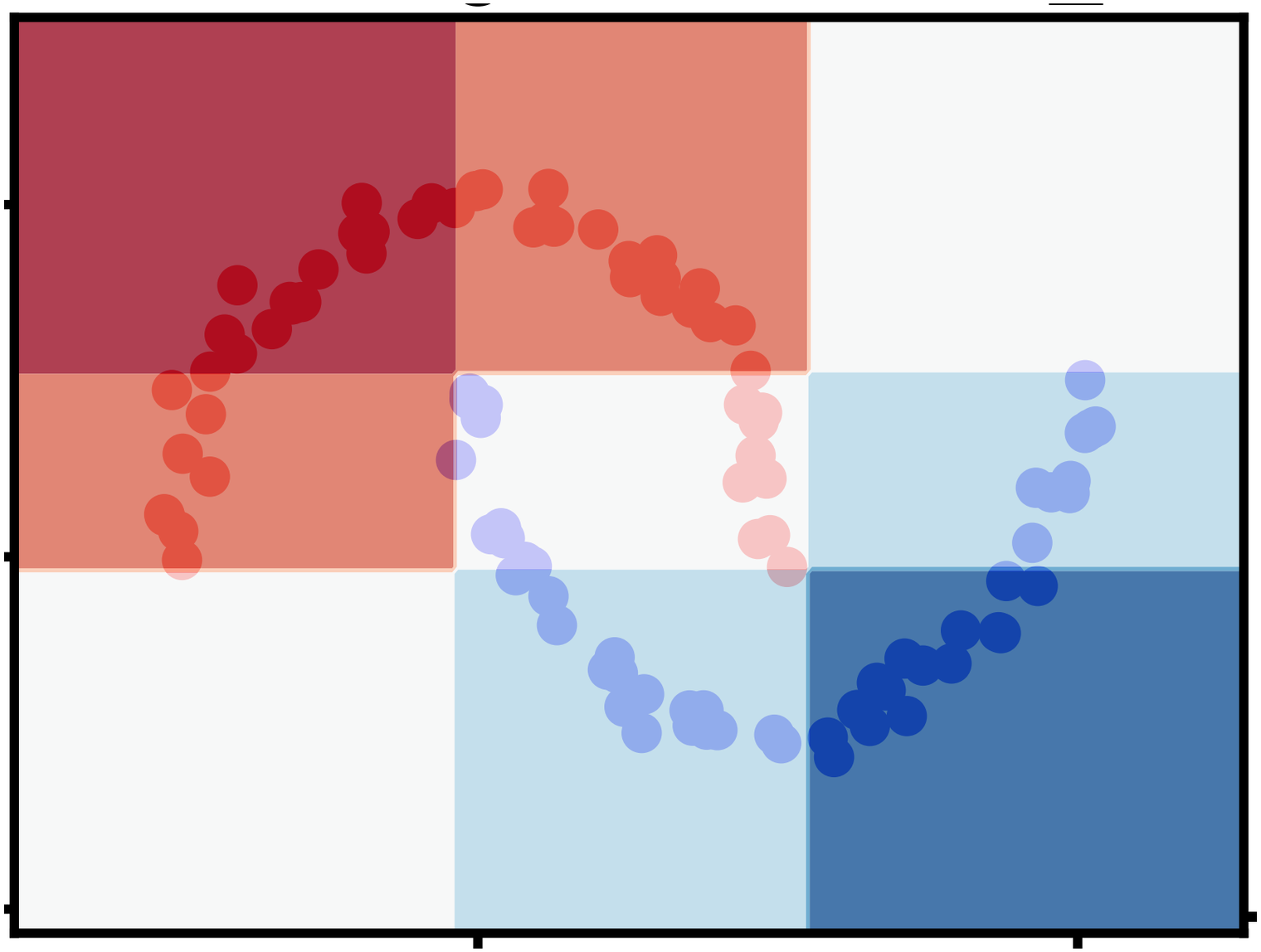
Weighted data 4



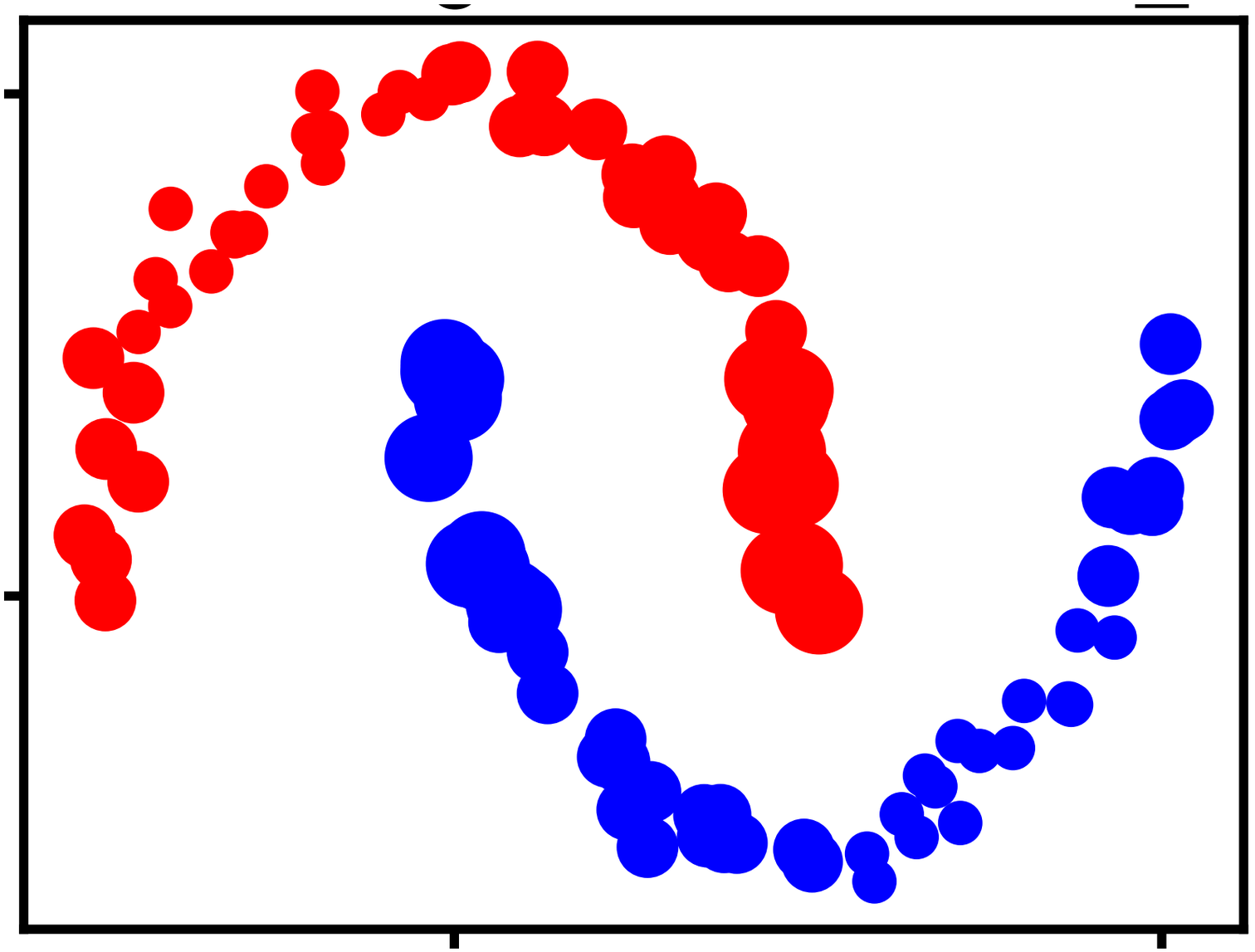
Weak classifier 4



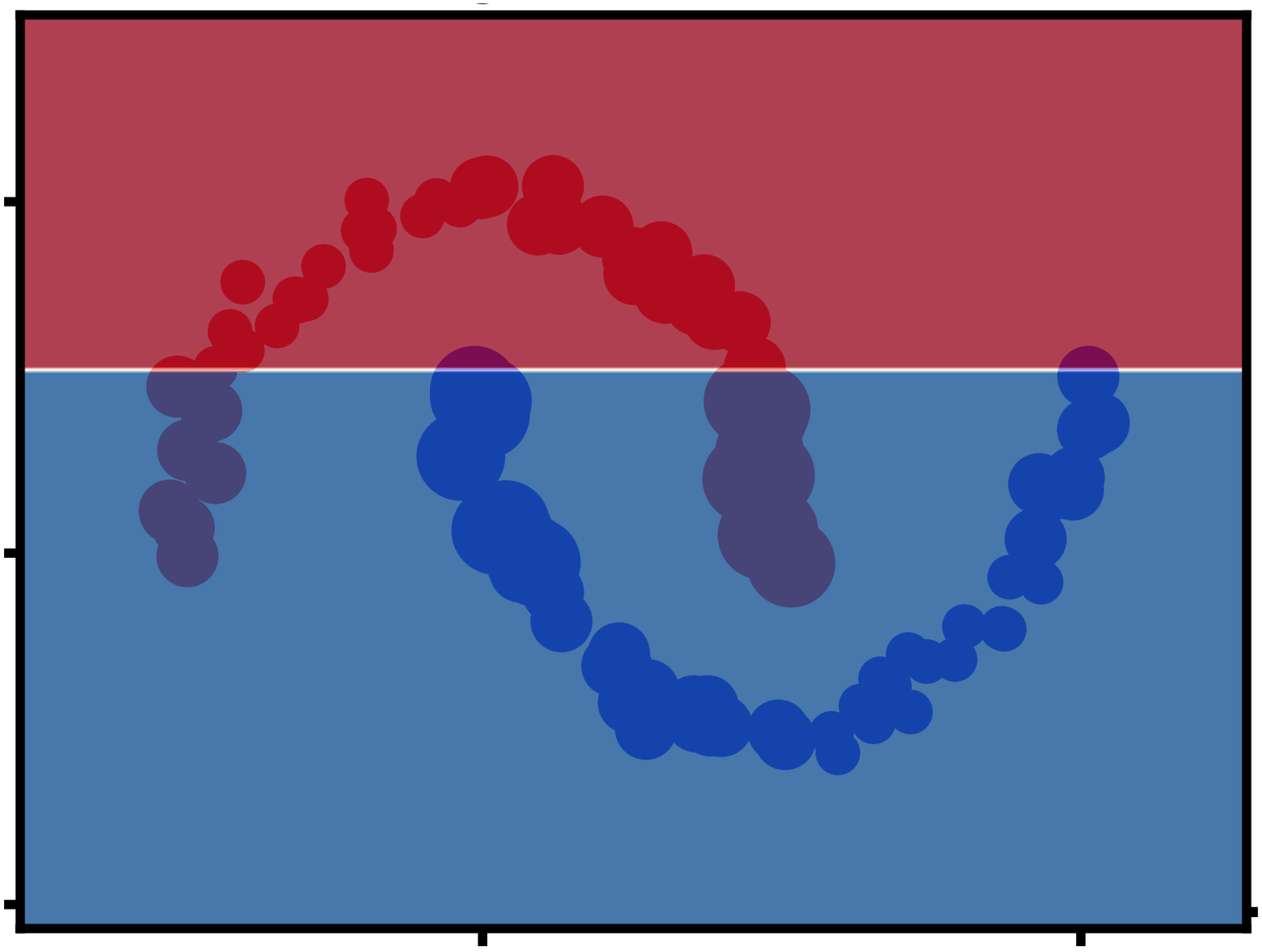
Strong classifier 4



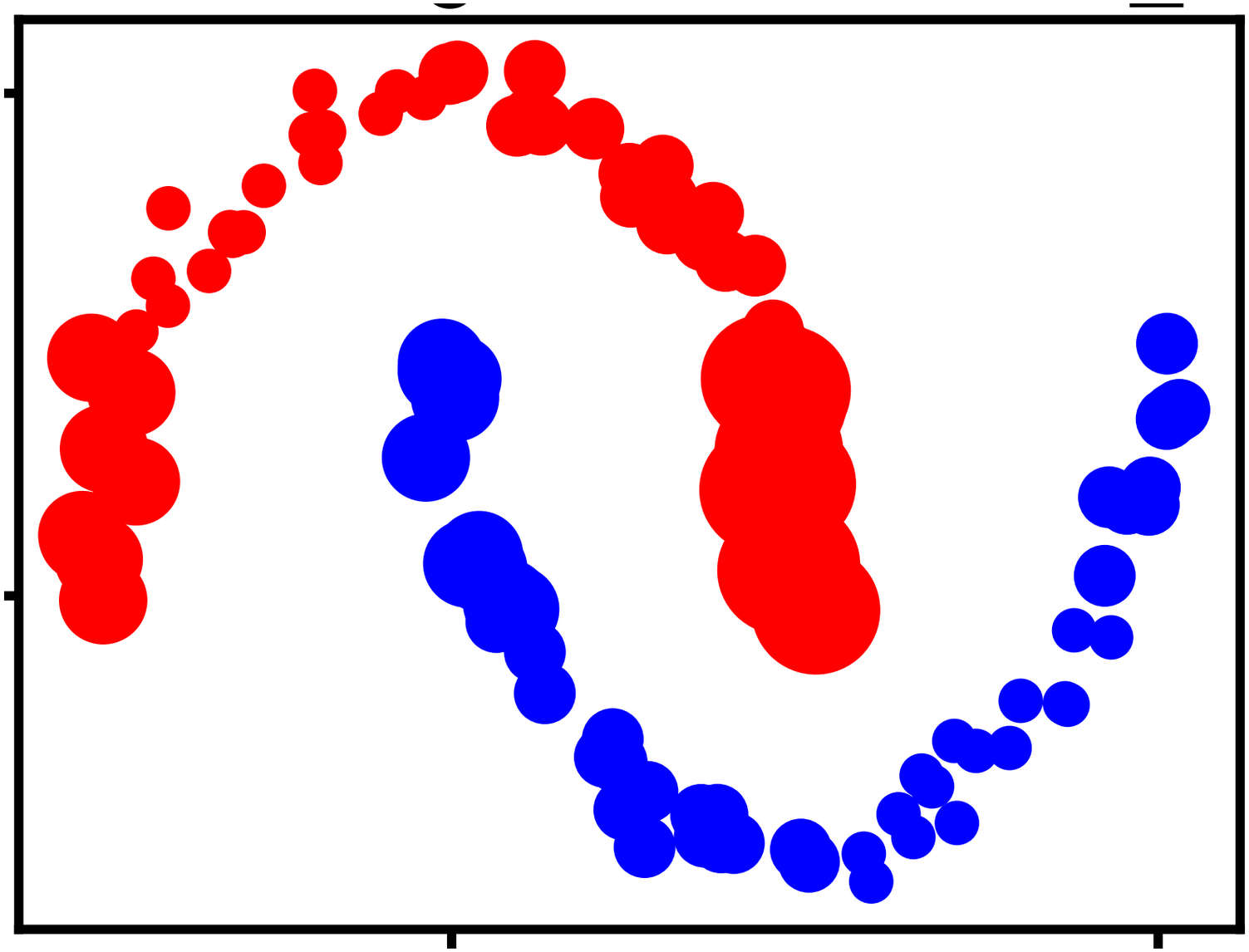
Weighted data 5



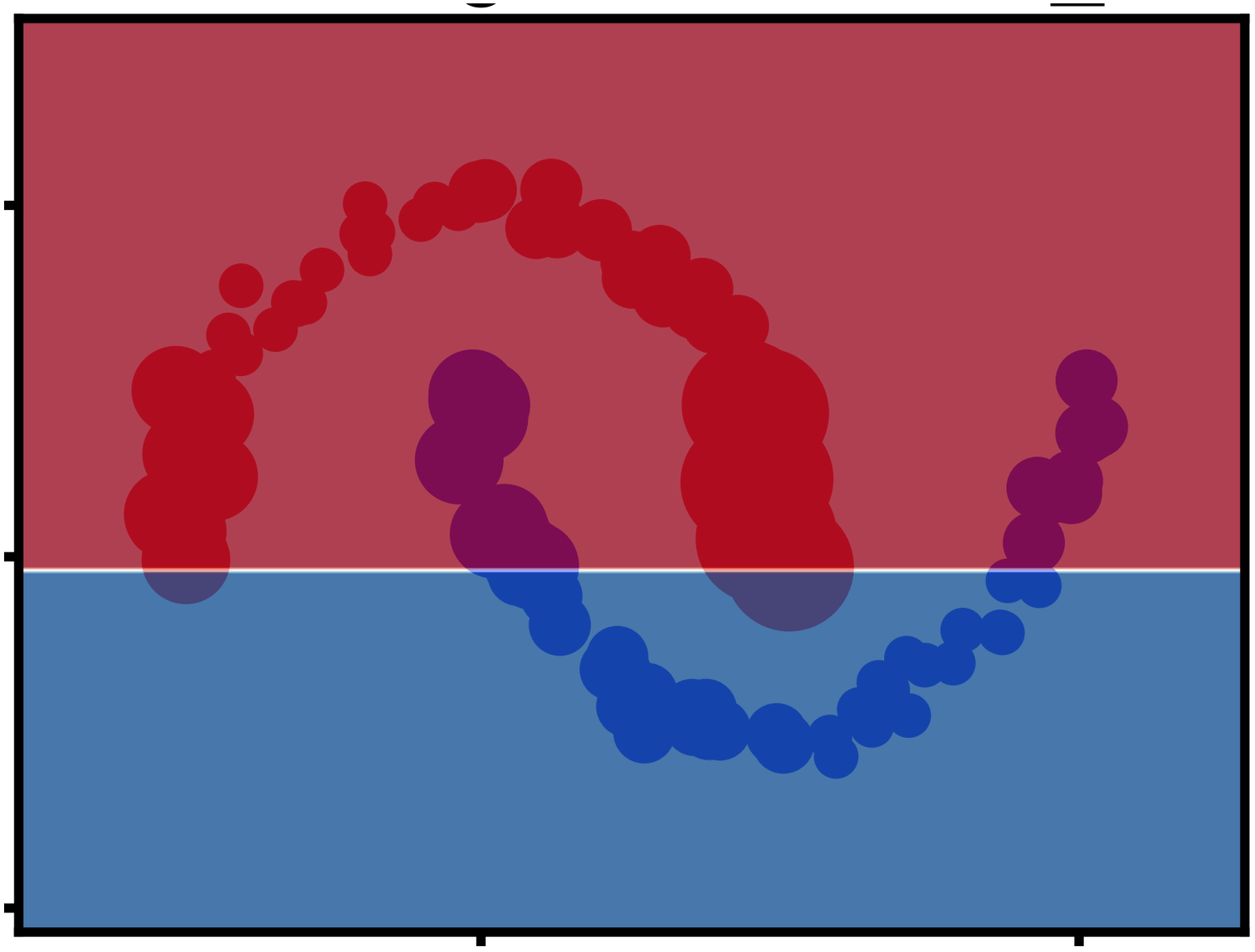
Weak classifier 5



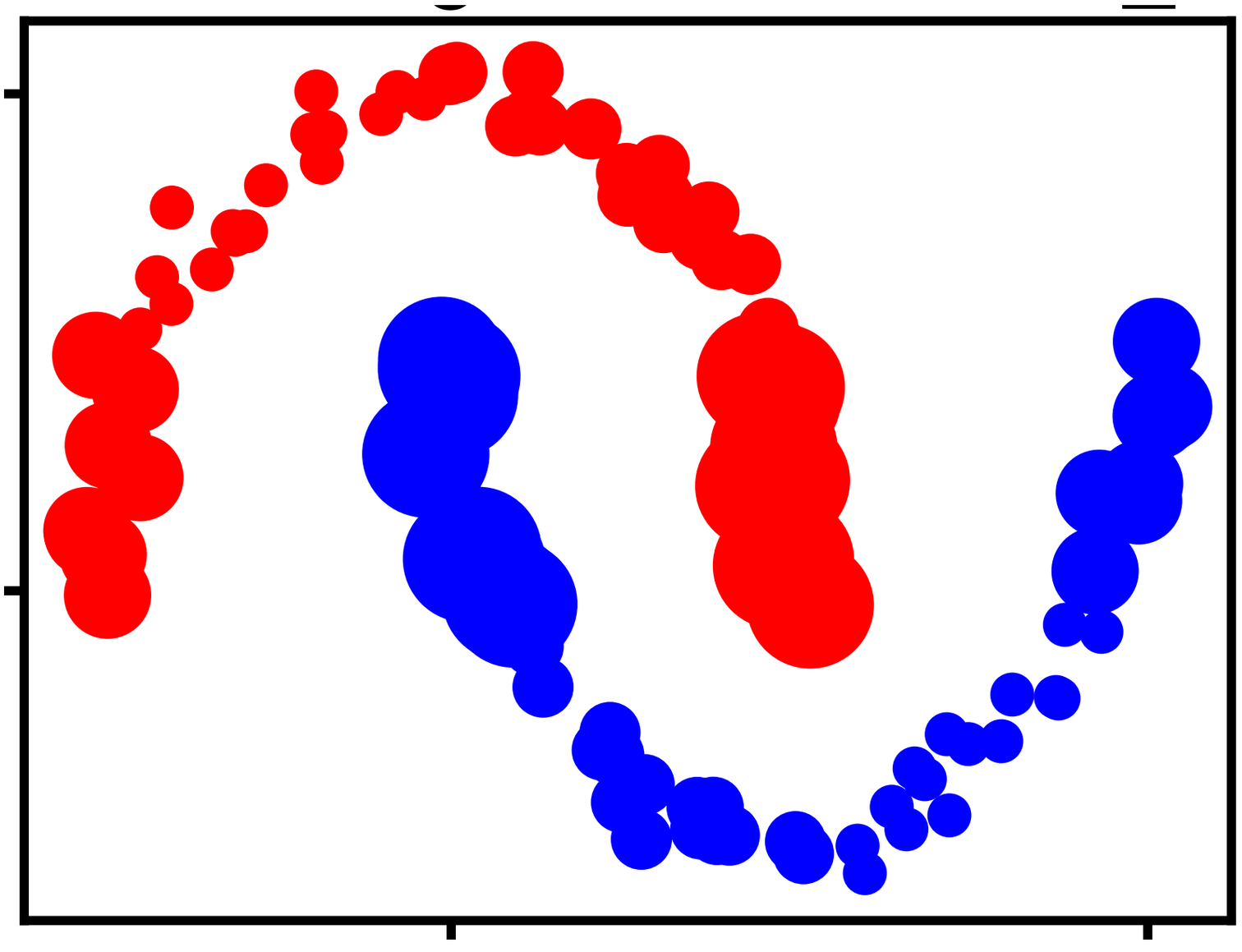
Weighted data 6



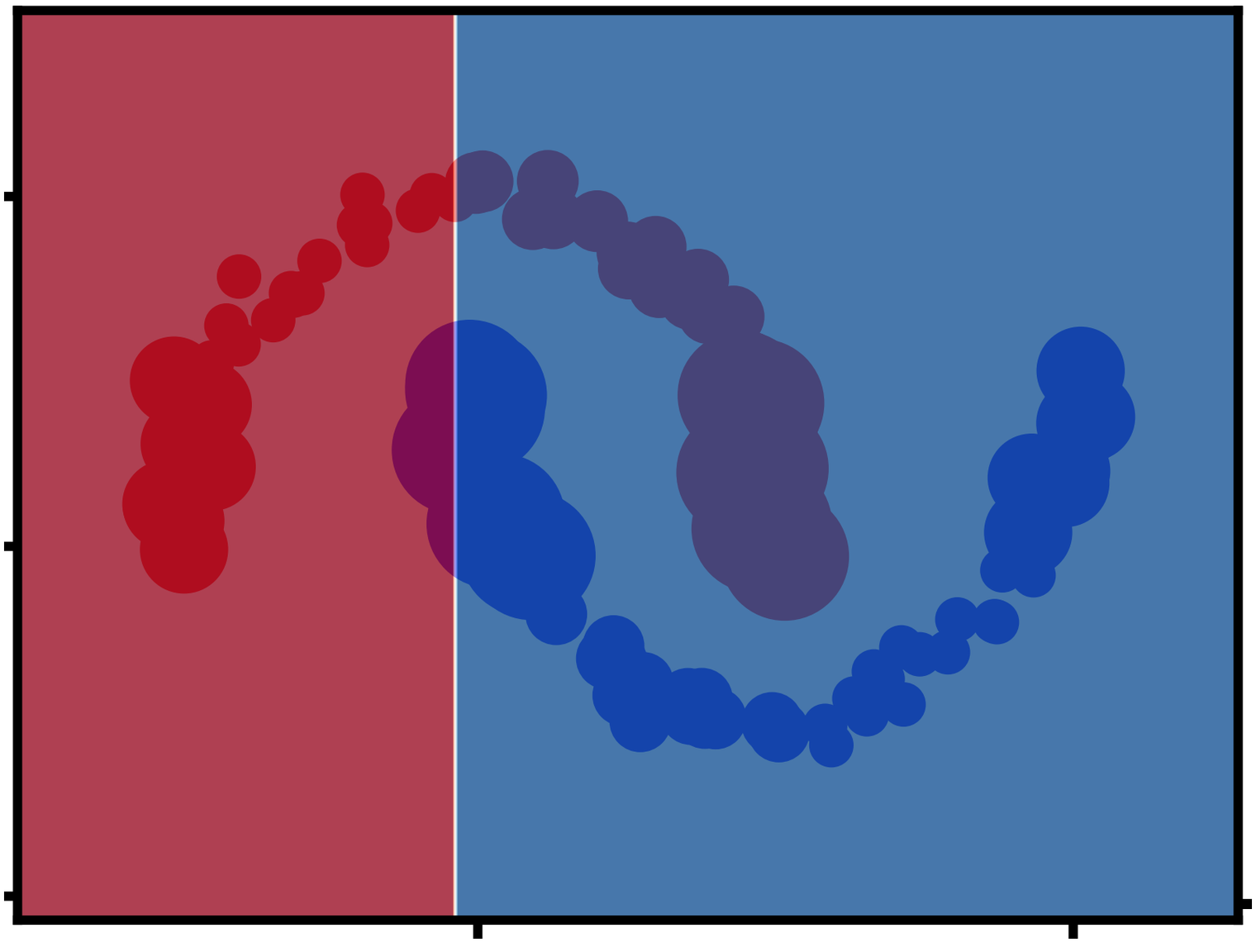
Weak classifier 6



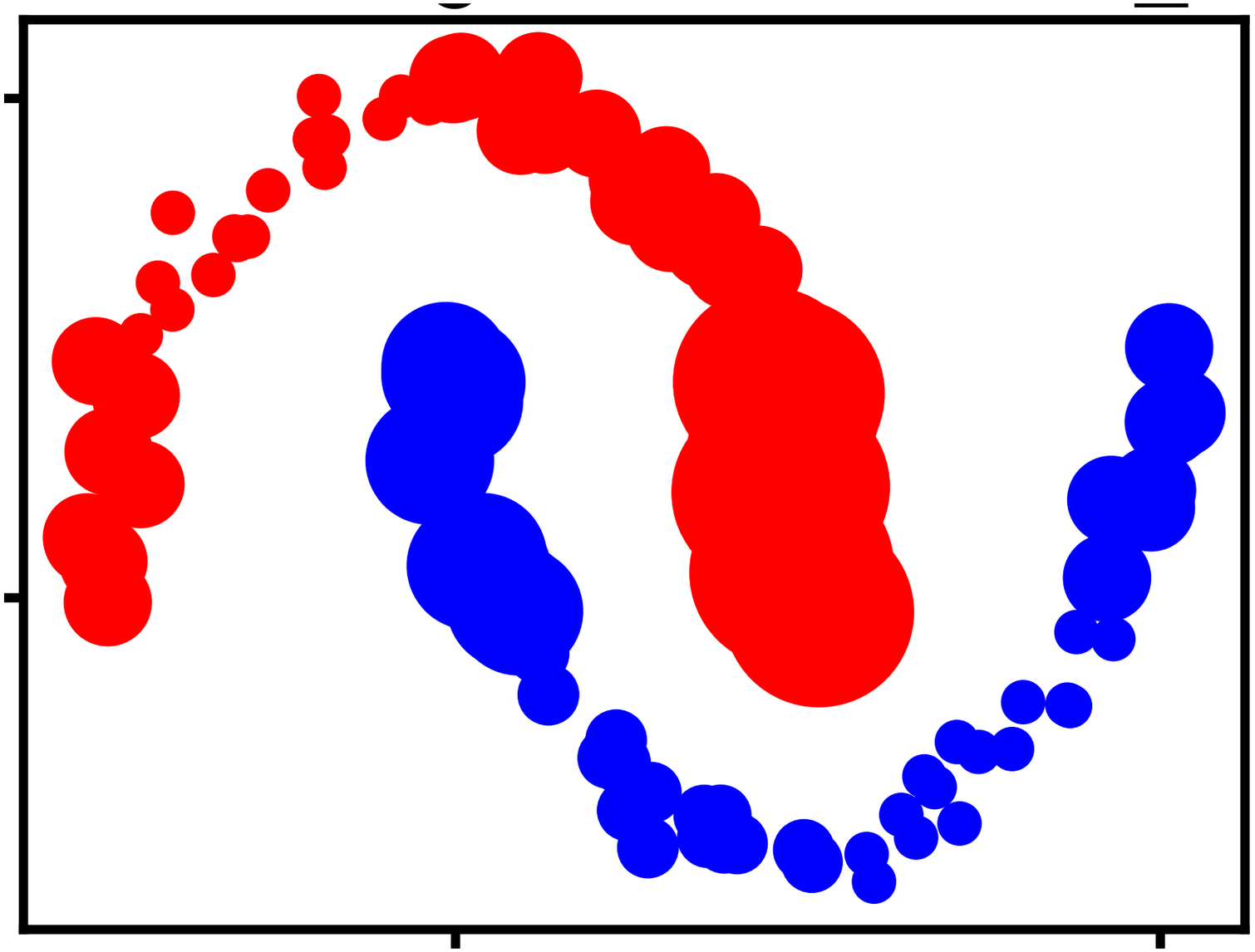
Weighted data 7



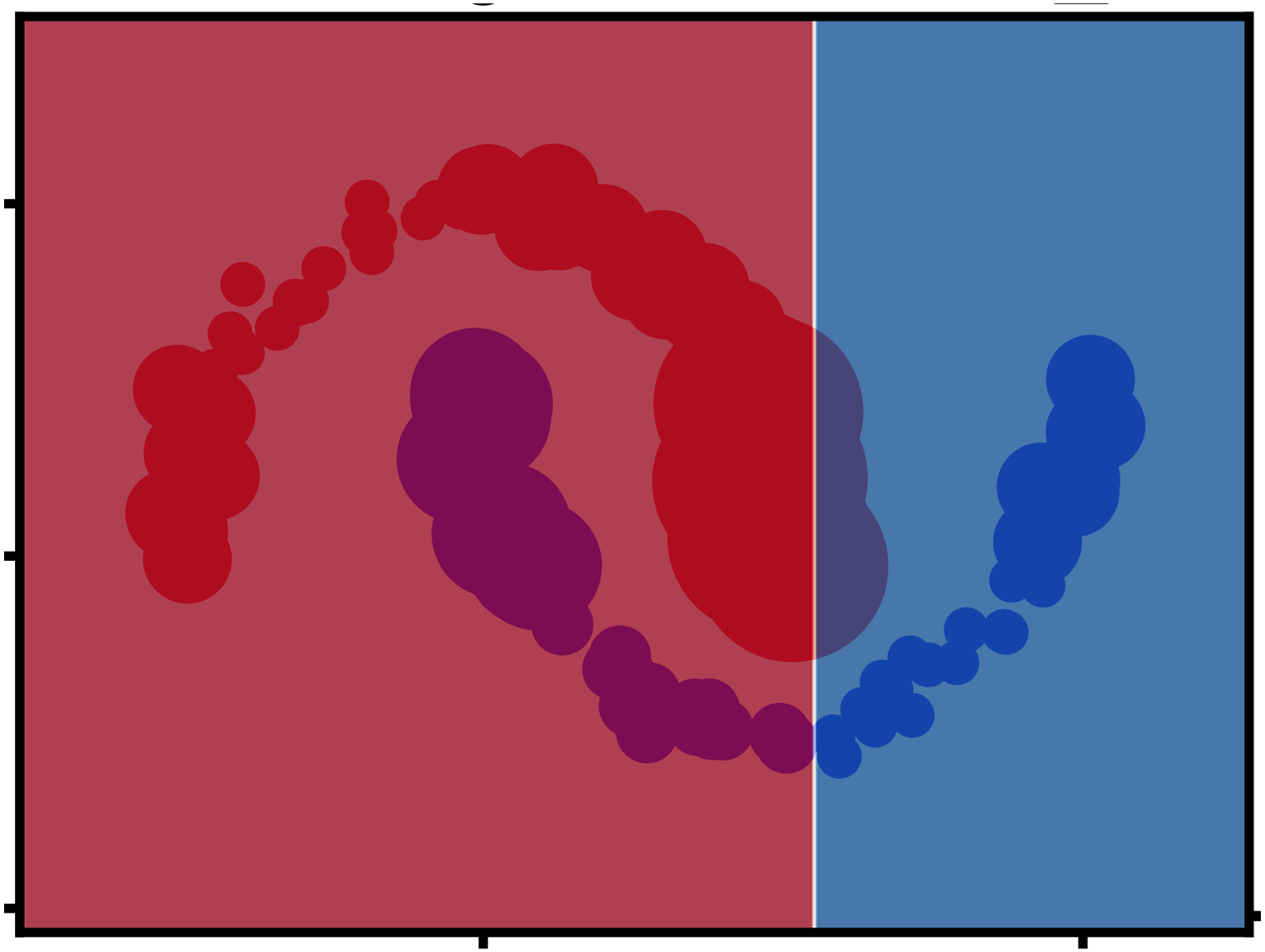
Weak classifier 7



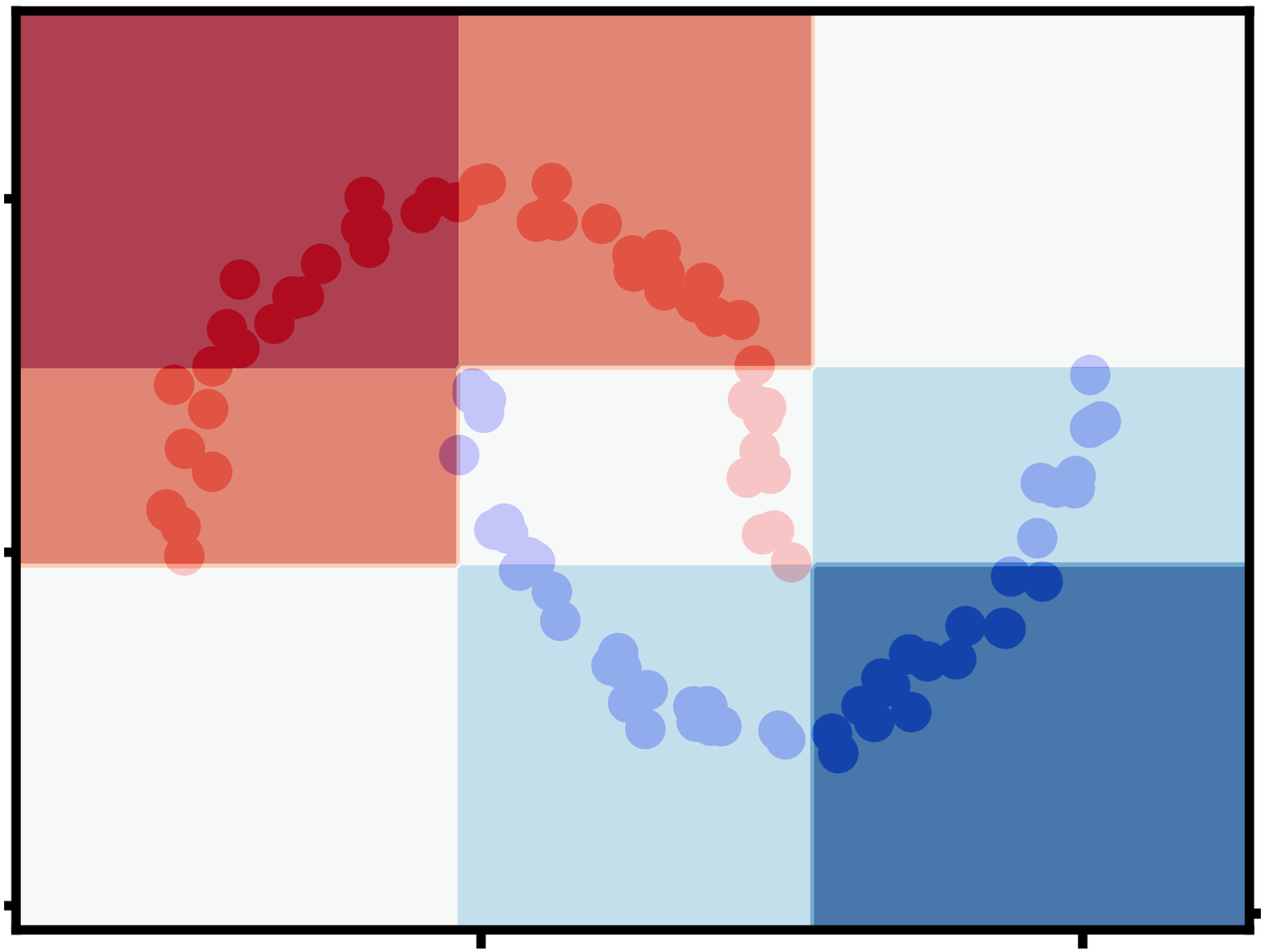
Weighted data 8



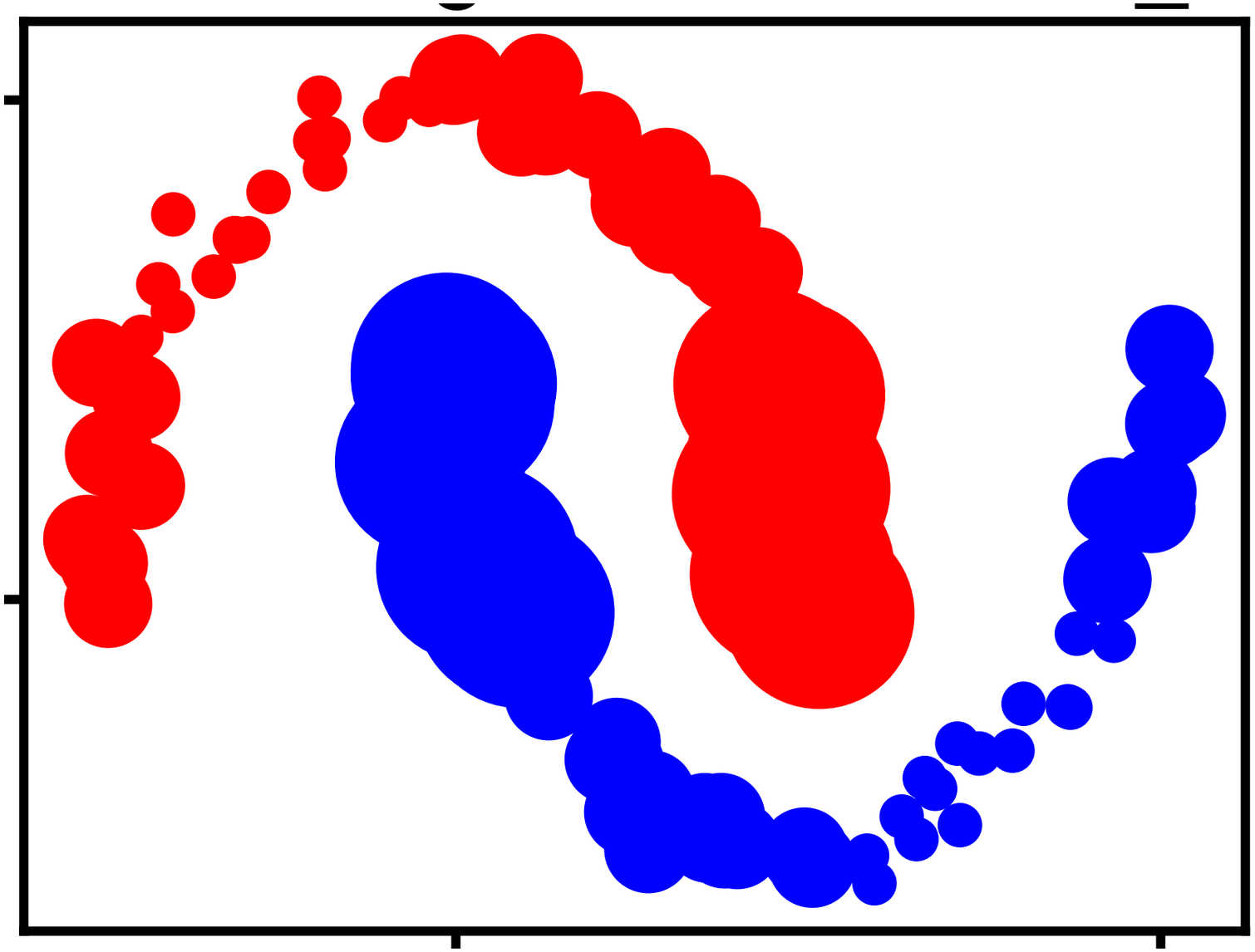
Weak classifier 8



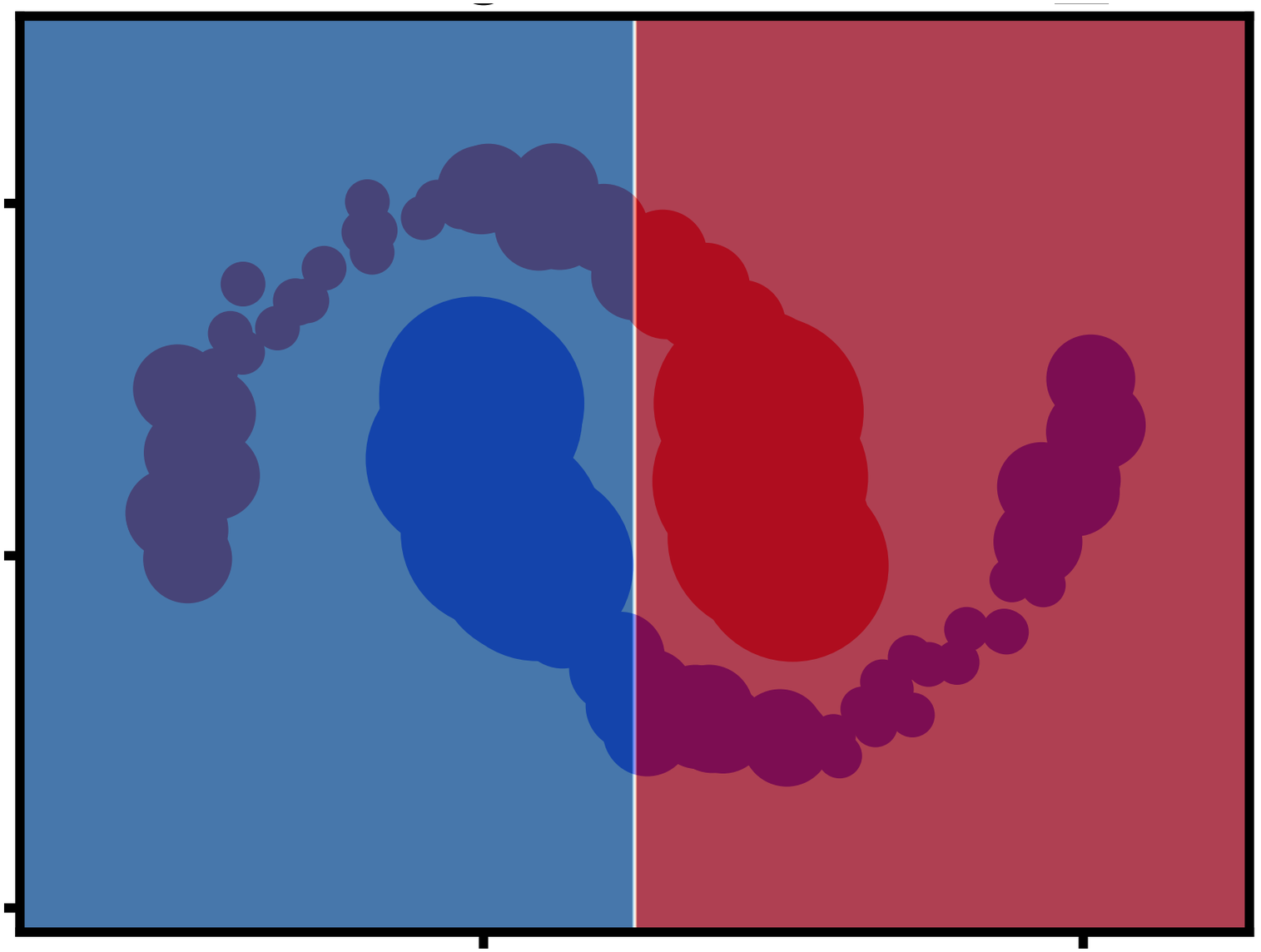
Strong classifier 8



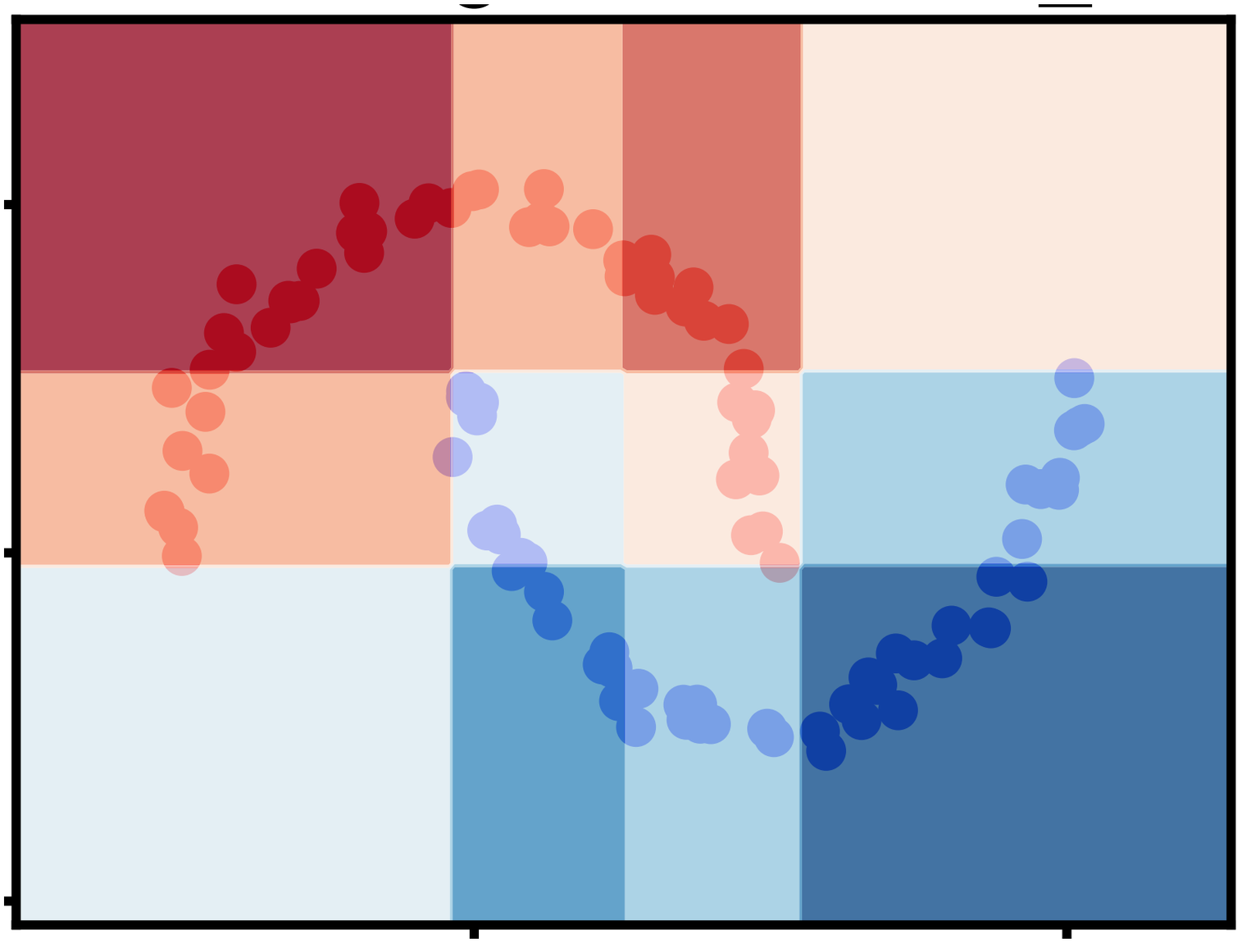
Weighted data 9



Weak classifier 9



Strong classifier 9



Boost by Majority

Input:

Weak learner with accuracy greater than $0.5 + \gamma$.
Training sample $S = (x_1, y_1), \ldots, (x_m, y_m) \in \mathcal{X} \times \{-1, +1\}$

$\beta = \frac{0.5+\gamma}{0.5-\gamma}$, $h_t : \mathcal{X} \to \{-1, +1\}$

Algorithm:

$\mathbf{w}^1 = \mathbf{1}$
For $t = 1$ to T:
$h_t = \text{WeakLearner}(S, \mathbf{w}^t)$
$l_{t,i} = |h_t(x_i) - y_i|/2$
$w_i^{t+1} = w_i^t \beta^{l_{t,i}}$
End For
Output: $h(x) = \text{sign}\left(\sum_{t=1}^{T} (h_t(x))\right)$

Convergence Proof Sketch

(Use as rough space)

Convergence Proof Sketch

Key idea: Consider $Z_t = \sum_{i=1}^{m} w_i^t$ Z_T = \sum Z_t

If h makes M mistakes, at least $T/2$ weak classifiers made mistakes on the same points.

The weight of a sample goes up by a factor of β in step t if mistake is made by the weak classifier.

The weight of a sample stays the same otherwise.

Thus,

$$Z_{T+1} \geq M\beta^{T/2}$$

Convergence Proof Sketch

(Use as rough space)

Convergence Proof Sketch

$$
\begin{aligned}
Z_{t+1} &= \sum_{i=1}^{m} w_i^{t+1} \\
&= \sum_{i \in A} w_i^t + \sum_{i \notin A} \beta w_i^t \\
&\leq (0.5+\gamma) \sum_i w_i^t + \beta(0.5-\gamma) \sum_i w_i^t \\
&= (1+2\gamma) Z_t
\end{aligned}
$$

(A=set of samples h_t got correctly.)

Thus,

$$Z_{T+1} \leq m(1+2\gamma)^T$$

Convergence Proof Sketch

(Use as rough space)

Convergence Proof Sketch

$$M\beta^{T/2} \leq Z_{T+1} \qquad Z_{T+1} \leq m(1+2\gamma)^T$$

$$M \leq m \cdot ((1+2\gamma)(1-2\gamma))^{T/2}$$

If $\gamma > 0$, the number of mistakes eventually goes to 0

Issues with Boost by Majority

- Requires a bound on γ : i.e how good the weak classifiers are in advance.
- If γ is set too small the algorithm converges slowly.
- If γ is set too high, the convergence proof doesn't hold and the boosted classifier could fail

AdaBoost

Algorithm:

$\mathbf{w}^1 = \mathbf{1}$

For $t = 1$ to T:

$h_t = \text{WeakLearner}(S, \mathbf{w}^t)$

$\gamma_t = \frac{1}{2} - \frac{1}{\sum_i w_i^t} \sum_{i=1}^{m} w_i^t \frac{|h_t(x_i) - y_i|}{2}$

$\beta_t = (0.5 + \gamma_t)/(0.5 - \gamma_t)$

$l_{t,i} = |h_t(x_i) - y_i|/2$

$w_i^{t+1} = w_i^t \beta_t^{l_{t,i}}$

End For

Output: $h(x) = \text{sign}\left(\sum_{t=1}^{T} (\log(\beta_t) h_t(x))\right)$

AdaBoost: Extensions and Generalisations

- Boosting can be extended to work with:
  - Learners that output a probability of a class.
  - Multiclass learners
  - Other reweighting schemes more robust to outliers (logitboost etc.)

References:

- Bagging: Leo Breiman (1984)
- Random forests: Leo Breiman (2001)
- Boosting : Freund and Schapire: (1998)