# Boosting for Comparison-Based Learning

Michaël Perrot[1] and Ulrike von Luxburg[1,2]

[1] *Max-Planck-Institute for Intelligent Systems, Max-Planck-Ring 4, 72076 Tübingen, Germany*
[2] *University of Tübingen, Department of Computer Science, Sand 14, 72076 Tübingen, Germany*

**Abstract**

We consider the problem of classification in a comparison-based setting: given a set of objects, we only have access to triplet comparisons of the form *object $x_i$ is closer to object $x_j$ than to object $x_k$*. In this paper we introduce TripletBoost, a new method that can learn a classifier just from such triplet comparisons. The main idea is to aggregate the triplets information into weak classifiers, which can subsequently be boosted to a strong classifier. Our method has two main advantages: (i) it is applicable to data from any metric space, and (ii) it can deal with large scale problems using only passively obtained and noisy triplets. We derive theoretical generalization guarantees and a lower bound on the number of necessary triplets, and we empirically show that our method is both competitive with state of the art approaches and resistant to noise.

## 1 Introduction

In the past few years the problem of comparison-based learning has attracted growing interest in the machine learning community (Agarwal et al., 2007; Jamieson and Nowak, 2011; Tamuz et al., 2011; Van Der Maaten and Weinberger, 2012; Heikinheimo and Ukkonen, 2013; Terada and von Luxburg, 2014; Amid and Ukkonen, 2015; Kleindessner and Luxburg, 2015; Jain et al., 2016; Haghiri et al., 2017). The motivation is to relax the assumption that an explicit representation of the objects or a distance metric between pairs of examples are available. Instead one only has access to a set of ordinal distance comparisons that can take several forms depending on the problem at hand. In this paper we focus on triplet comparisons of the form *object $x_i$ is closer to object $x_j$ than to object $x_k$*, that is on relations of the form $d(x_i, x_j) < d(x_i, x_k)$ where $d$ is an unknown metric[1].

We address the problem of classification with noisy triplets that have been obtained in a passive manner: the examples lie in an unknown metric space, not necessarily Euclidean, and we are only given a small set of triplet comparisons — there is no way in which we could actively ask for more triplets. Furthermore we assume that the answers to the triplet comparisons can be noisy. To deal with this problem one can try to first recover an explicit representation of the examples, a task that can be solved by ordinal embedding approaches (Agarwal et al., 2007; Van Der Maaten and Weinberger, 2012; Terada and von Luxburg, 2014; Jain et al., 2016), and then apply standard machine learning approaches. However, such embedding methods assume that the examples lie in a Euclidean space and do not scale well with the number of examples: typically they are too slow for datasets with more than $10^3$ examples. As an alternative, it would be desirable to have a classification algorithm that can work with triplets directly, without taking a detour via ordinal embedding. To the best of our knowledge, for the case of passively obtained triplets, this problem has not yet been solved in the literature.

Another interesting question in this context is that of the minimal number of triplets required to successfully learn a classifier. It is known that to exactly recover an ordinal embedding one needs

[1] Note that this kind of ordinal information is sometimes just used as side information (Bellet et al., 2015; Kane et al., 2017), but, as the references in the main text, we focus on the setting where ordinal comparisons are the sole information available.

of the order $\Omega(n^3)$ passively queried triplets in the worst case (essentially all of them), unless we make stronger assumptions on the underlying metric space (Jamieson and Nowak, 2011). However, classification is a problem which seems simpler than ordinal embedding, and thus it might be possible to obtain better lower bounds.

In this paper we propose TripletBoost, a method for classification that is able to learn using only passively obtained triplets while not making any assumptions on the underlying metric space. To the best of our knowledge this is the first approach that is able to solve this problem. Our method is based on the idea that the triplets can be aggregated into simple *triplet classifiers*, which behave like decision stumps and are well-suited for boosting approaches (Schapire and Freund, 2012). From a theoretical point of view we prove that our approach learns a classifier with low training error, and we derive generalization guarantees that ensure that its error on new examples is bounded. Furthermore we derive a new lower bound on the number of triplets that are necessary to ensure useful predictions. From an empirical point of view we demonstrate that our approach can be applied to datasets that are several order of magnitudes larger than the ones that can currently be handled by ordinal embedding methods. Furthermore we show that our method is quite resistant to noise.

# 2 The TripletBoost Algorithm

In this paper we are interested in multi-class classification problems. Let $(\mathcal{X}, d)$ be an unknown and general metric space, typically not Euclidean. Let $\mathcal{Y}$ be a finite label space. Let $S = \{(x_i, y_i)\}_{i=1}^n$ be a set of $n$ examples drawn i.i.d. from an unknown distribution $\mathcal{D}_S$ defined over $\mathcal{X} \times \mathcal{Y}$. Note that we use the notation $x_i$ as a convenient way to identify an object; it does not correspond to any explicit representation that could be used by an algorithm (such as coordinates in a vector space). Let $T = \{(x_i, x_j, x_k) : (x_i, y_i), (x_j, y_j), (x_k, y_k) \in S, x_j \neq x_k\}$ be a set of $m$ triplets. Each ordered tuple $(x_i, x_j, x_k) \in T$ encodes the following relation between the three examples:

$$d(x_i, x_j) < d(x_i, x_k)\,. \tag{1}$$

Given the triplets in $T$ and the label information of all points, our goal is to learn a classifier. We make two main assumptions about the data. First we assume that the triplets are uniformly and independently sampled from the set of all possible triplets. Second, we assume that the triplets in $T$ can be noisy (in the sense that the inequality has been swapped, for example $(x_i, x_k, x_j) \in T$ while the true relation is $d(x_i, x_j) < d(x_i, x_k)$), but the noise is uniform and independent from one triplet to another. In the following $\mathbb{I}_a$ denotes the indicator function returning 1 when a property $a$ is verified and 0 otherwise, $\mathcal{W}_c$ is an empirical distribution over $S \times \mathcal{Y}$, and $w_{c,x_i,y}$ is the weight associated to object $x_i$ and label $y$.

## 2.1 Weak Triplet Classifiers

Rather than considering triplets as individual pieces of information we propose to aggregate them into decision stumps that we call *triplet classifiers*. The underlying idea is to select two reference examples, $x_j$ and $x_k$, and to divide the space into two *half-spaces*: examples that are closer to $x_j$ and examples that are closer to $x_k$. This is illustrated in Figure 1. In principle, this can be achieved with triplets only. However, a major difficulty in our setting is that our triplets are passively obtained: for most training points $x_i$ we do not know whether they are closer to $x_j$ or $x_k$. In particular, it is impossible to evaluate the classification accuracy of such a simple classifier on the whole training set. To deal with this problem we propose to use an abstention scheme where a triplet classifier abstains if it does not know on which side of the hyperplane the considered point lies. Given a set $T$ of triplets and two reference points $x_j$ and $x_k$ we formally define a triplet classifier as follows:

$$h_{j,k}(x) = \begin{cases} o_j & \text{if } (x, x_j, x_k) \in T, \\ o_k & \text{if } (x, x_k, x_j) \in T, \\ \vartheta & \text{otherwise.} \end{cases}$$

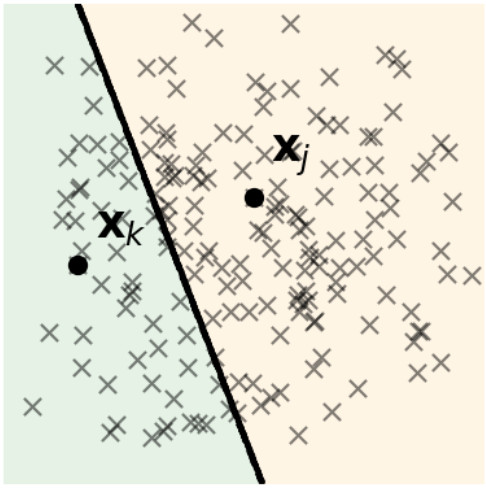



Figure 1: Example of a triplet classifier. Given two reference points $x_j$ and $x_k$, the space is divided in two half-spaces: examples closer to $x_j$ and examples closer to $x_k$. Triplet information is enough to reveal which point $x_i$ is in which half-space.

In our multi-class setting, $o_j$ and $o_k$ will be sets of labels, that is $o_j, o_k \subseteq \mathcal{Y}$. In Section 2.2 we describe how we choose them in a data dependent fashion to obtain classifiers with minimal error on the training set. The prediction $\vartheta$ simply means that the triplet classifier abstains on the example.

Let $\mathcal{H}$ be the set of all possible triplet classifiers. Given $n$ examples, there is $n(n-1)/2$ ways to choose the pair $(x_j, x_k)$ and $2^{|\mathcal{Y}|}$ possible sets for both $o_j$ and $o_k$. This implies that $|\mathcal{H}| = \frac{n(n-1)}{2} 2^{2|\mathcal{Y}|}$. These triplet classifiers are very simple and, in practice, we do not expect them to perform well at all. But we prove in Section 3.1 that, for appropriate choices of $o_j$ and $o_k$, they are at least as good as random predictors. This is all we need to ensure that they can be used successfully in a boosting framework. The next section describes how this works.

## 2.2 TripletBoost

Boosting is based on the insight that weak classifiers (that is, classifiers marginally better than random predictors) are usually easy to obtain and can be combined in a weighted linear combination to obtain a strong classifier. This weighted combination can be obtained in an iterative fashion where, at each iteration, a weak-classifier is chosen and weighted so as to minimize the error on the training examples. The weights of the points are then updated to put more focus on hard-to-classify examples (Schapire and Freund, 2012). In this paper we use a well-known boosting algorithm called AdaBoost.MO (Schapire and Singer, 1999; Schapire and Freund, 2012). This method can handle multi-class problems with a one-against-all approach, works with abstaining classifiers and is theoretically well founded. We summarize the main steps of our approach in Algorithm 1 and detail them below.

**Choosing a triplet classifier.** To choose a triplet classifier we proceed in two steps. In the first step we select two reference points $x_j$ and $x_k$ such that $y_j \neq y_k$. This is done by randomly sampling from an empirical distribution $\mathcal{W}_{c,\mathcal{X}}$ on the examples. Here $\mathcal{W}_{c,\mathcal{X}}$ denotes the marginal distribution of $\mathcal{W}_c$ with respect to the examples. This distribution will be chosen iteratively to put more focus on those parts of the space that are hard to classify while promoting triplet classifiers that are able to separate different classes (see Equation 4 below).

In the second step we choose $o_j$ and $o_k$, the sets of labels that should be predicted for each half

**input :** $S = \{(x_i, y_i)\}_{i=1}^n$ a set of $n$ examples, $T = \{(x_i, x_j, x_k) : x_j \neq x_k\}$ a set of $m$ triplets.
**output:** $H(\cdot)$ a strong classifier.
**begin**
Let $\mathcal{W}_1$ be the empirical uniform distribution: $\forall (x_i, y_i) \in S, \forall y \in \mathcal{Y}, w_{1,x_i,y} = \frac{1}{n|\mathcal{Y}|}$.
**for** $c = 1, \ldots, C$ **do**
Choose a triplet classifier $h_c$ according to $\mathcal{W}_c$ (Equation (2)).
Compute the weight $\alpha_c$ of $h_c$ according to its performance on $\mathcal{W}_c$ (Equation (3)).
Update the weights of the examples to obtain a new distribution $\mathcal{W}_{c+1}$ (Equation (4)).
**end**
**return** $H(\cdot) = \arg\max_{y \in \mathcal{Y}} \left( \sum_{c=1}^{C} \alpha_c \mathbb{I}_{h_c(\cdot) \neq \vartheta \wedge y \in h_c(\cdot)} \right)$
**end**

**Algorithm 1:** TripletBoost: A boosting algorithm using triplet classifiers.

space of the triplet classifier. Given one of the half spaces, we propose to add a label to the set of predicted labels if the weight of examples of this class is greater than the weight of examples of different classes. Formally, with $w_{c,x_i,y}$ defined as in Algorithm 1, we construct $o_j$ as follows:

$$o_j = \left\{ y : \sum_{(x_i,y_i) \in S,\ (x_i,x_j,x_k) \in T} \mathbb{I}_{y=y_i} w_{c,x_i,y} > \sum_{(x_i,y_i) \in S,\ (x_i,x_j,x_k) \in T} \mathbb{I}_{y \neq y_i} w_{c,x_i,y} \right\}. \tag{2}$$

We construct $o_k$ in a similar way. The underlying idea is that adding $h_c$ to the current combination of triplet classifiers should improve the predictions on the training set as much as possible. In Section 3.1 we show that this strategy maximizes the accuracy of the triplet classifier on the training set and ensures that the selected triplet classifier is either a weak classifier or has a weight $\alpha_c$ of 0.

**Computing the weight of the triplet classifier.** To choose the weight of the triplet classifier $h_c$ we start by computing $W_{c,+}$ and $W_{c,-}$, the weights of correctly and incorrectly classified examples:

$$\begin{aligned} W_{c,+} &= \sum_{(x_i,y_i) \in S,\ h_c(x_i) \neq \vartheta} \left( \mathbb{I}_{y_i \in h_c(x_i)} w_{c,x_i,y_i} + \sum_{y \neq y_i} \mathbb{I}_{y \notin h_c(x_i)} w_{c,x_i,y} \right), \\ W_{c,-} &= \sum_{(x_i,y_i) \in S,\ h_c(x_i) \neq \vartheta} \left( \mathbb{I}_{y_i \notin h_c(x_i)} w_{c,x_i,y_i} + \sum_{y \neq y_i} \mathbb{I}_{y \in h_c(x_i)} w_{c,x_i,y} \right). \end{aligned} \tag{3}$$

We then set $\alpha_c = \frac{1}{2} \log \left( \frac{W_{c,+} + \frac{1}{n}}{W_{c,-} + \frac{1}{n}} \right)$. The term $\frac{1}{n}$ is a smoothing constant (Schapire and Singer, 2000): in our setting with few, passively queried triplets it helps to avoid numerical problems that

might arise when $W_{c,+} = 0$ or $W_{c,-} = 0$. In Theorem 2 we show that, with this choice of $\alpha_c$, the training error decreases as the number of iterations of our algorithm increases.

**Updating the weights of the examples.** In each iteration of our algorithm, a new triplet classifier $h_c$ is added to the weighted combination of classifiers, and we need to update the empirical distribution $\mathcal{W}_c$ over the examples for the next iteration. The idea is (i) to reduce the weights of correctly classified examples, (ii) to keep constant the weights of the examples for which the current triplet classifier abstains, and (iii) to increase the weights of incorrectly classified examples. The weights are then normalized by a factor $Z_c$ so that $\mathcal{W}_{c+1}$ remains an empirical distribution over the examples. Formally, $\forall (x_i, y_i) \in S, \forall y \in \mathcal{Y}$, if $h_c(x_i) = \vartheta$ then $w_{c+1,x_i,y} = \frac{w_{c,x_i,y}}{Z_c}$ and if $h_c(x_i) \neq \vartheta$ then

$$w_{c+1,x_i,y} = \frac{w_{c,x_i,y}}{Z_c} \exp\left\{-\alpha_c \left(\mathbb{I}_{y=y_i} - \mathbb{I}_{y\neq y_i}\right) \times \left(\mathbb{I}_{y\in h_c(x_i)} - \mathbb{I}_{y\notin h_c(x_i)}\right)\right\}. \tag{4}$$

**Using $H$ for prediction.** Given a new example $x$, TripletBoost predicts its label as

$$H(x) = \arg\max_{y\in\mathcal{Y}} \left( \sum_{c=1}^{C} \alpha_c \mathbb{I}_{h_c(x)\neq\vartheta \wedge y\in h_c(x)} \right) \tag{5}$$

that is the label with the highest weight as predicted by the weighted combination of selected triplet classifiers. However, recall that we are in a passive setting, and thus we assume that we are given a set $T_x = \{(x, x_j, x_k) : (x_j, y_j), (x_k, y_k) \in S, x_j \neq x_k\}$ of triplets associated with the example $x$ (but there is no way to choose them). Hence, some of the triplets in $T_x$ correspond to triplet classifiers in $H$ (that is $(x, x_j, x_k) \in T_x$ and the reference points for $h_c$ were $x_j$ and $x_k$) and some do not. In particular, it might happen that none of the triplets in $T_x$ corresponds to a triplet classifier in $H$ and, in this case, $H$ can only randomly predict a label. In Section 3.3 we provide a lower bound on the number of triplets necessary to avoid this behaviour. The main computational bottleneck when predicting the label of a new example $x$ is to check whether the triplets in $T_x$ match a triplet classifier in $H$. A naive implementation would compare each triplet in $T_x$ to each triplet classifier, which can be as expensive as $O(|T_x|\, C)$. Fortunately, by first sorting the triplets and the triplet classifiers, a far more reasonable complexity of $O(|T_x| \log(|T_x|) + C \log(C))$ can be achieved.

# 3 Theoretical Analysis

In this section we show that our approach is theoretically well founded. First we prove that the triplet classifiers with non-zero weights are weak learners: they are slightly better than random predictors (Theorem 1). Building upon this result we show that, as the number of iterations increases, the training error of the strong classifier learned by TripletBoost is decreased (Theorem 2). Then, to ensure that TripletBoost does not over-fit, we derive a generalization bound showing that, given a sufficient amount of training examples, the test error is bounded (Theorem 3). Finally, we derive a lower bound on the number of triplets necessary to ensure that TripletBoost does not learn a random predictor (Theorem 4). For the sake of readability we defer the proofs of all the theorems to the supplementary material. In the main paper we just provide some proof sketches.

## 3.1 Triplet Classifiers and Weak Learners

We start this theoretical analysis by showing that our strategy to choose the predicted labels of the triplet classifiers in Equation (2) is optimal: it ensures that their error is minimal on the training set (compared to any other labelling strategy). We also show that the triplet classifiers are never worse than random predictors and in fact, that only those triplets classifiers that are

weak classifiers (strictly better than random classifiers) are affected a non-zero weight. This is summarized in the next theorem.

**Theorem 1 (Triplet classifiers and weak learners).** *Let $\mathcal{W}_c$ be an empirical distribution over $S \times \mathcal{Y}$ and $h_c$ be the corresponding triplet classifier chosen as described in Section 2.2. It holds that:*

1. *the error of $h_c$ on $\mathcal{W}_c$ is at most the error of a random predictor and is minimal compared to other labelling strategies,*
2. *the weight $\alpha_c$ of the classifier is non-zero if and only if, $h_c$ is a weak classifier, that is is strictly better than a random predictor.*

*Proof sketch.* The first part of the theorem follows by noting that our labelling strategy minimizes $W_{c,-}$, the weights of incorrectly classified examples, and thus the error of $h_c$ on $\mathcal{W}_c$. The second part of the theorem is due to the definition of the weights $\alpha_c$. □

## 3.2 Boosting guarantees

From a theoretical point of view the boosting framework has been extensively investigated and it has been shown that most AdaBoost-based methods decrease the training error at each iteration (Freund and Schapire, 1997). Another question that has attracted a lot of attention is the problem of generalization. It is known that when the training error has been minimized, AdaBoost-based methods often do not over-fit and it might even be beneficial to further increase the number of weak learners. A popular explanation is the margin theory which says that as the number of iterations increases, the confidence of the algorithm in its predictions increases and thus, the test accuracy is improved (Schapire et al., 1998; Breiman, 1999; Wang et al., 2011; Gao and Zhou, 2013). TripletBoost is based on AdaBoost.MO, and thus it inherits the theoretical guarantees presented above. In this section, we provide two theorems which show (i) that TripletBoost reduces the training error as the number of iterations increases, and (ii) that it generalizes well to new examples.

The following theorem shows that, as the number of iterations increases, TripletBoost decreases the training error.

**Theorem 2 (Reduction of the Training Error).** *Let $S$ be a set of $n$ examples and $T$ be a set of $m$ triplets (obtained as described in Section 2). Let $H(\cdot)$ be the classifier obtained after $C$ iterations of TripletBoost (Algorithm 1) using $S$ and $T$ as input. It holds that:*

$$\mathop{\mathbb{P}}_{(x,y)\in S}[H(x) \neq y] \leq \frac{|\mathcal{Y}|}{2} \prod_{c=1}^{C} Z_c$$

*with* $Z_c = (1 - W_{c,+} - W_{c,-}) + (W_{c,+}) \cdot \sqrt{\frac{W_{c,-}+\frac{1}{n}}{W_{c,+}+\frac{1}{n}}} + (W_{c,-}) \cdot \sqrt{\frac{W_{c,+}+\frac{1}{n}}{W_{c,-}+\frac{1}{n}}} \leq 1.$

*Proof sketch.* The proof of this theorem relies on the expansion of the weights of the examples in Algorithm 1 and on the result obtained in Theorem 1. □

The next theorem shows that the true error of a classifier learned by TripletBoost can be bounded by a quantity related to the confidence of the classifier on the training examples, with respect to a margin, plus a term which decreases as the number of examples increases. The confidence of the classifier on the training examples is also bounded and decreases for sufficiently small margins.

**Theorem 3 (Generalization Guarantees).** *Let $\mathcal{D}_{\mathcal{S}}$ be a distribution over $\mathcal{X} \times \mathcal{Y}$, let $S$ be a set of $n$ examples drawn i.i.d. from $\mathcal{D}_{\mathcal{S}}$, and let $T$ be a set of $m$ triplets (obtained as described in Section 2). Let $H(\cdot)$ be the classifier obtained after $C$ iterations of TripletBoost (Algorithm 1)*

*using $S$ and $T$ as input. Let $\mathcal{H}$ be a set of triplet classifiers as defined in Section 2.1. Then, with probability at least $1-\delta$, we have*

$$\mathop{\mathbb{P}}_{(x,y)\sim\mathcal{D}_S}[H(x)\neq y] \leq \mathop{\mathbb{P}}_{(x,y)\in S}[\theta_{f,\eta}(x,y)\leq\theta] + \mathcal{O}\left(\sqrt{\frac{\log\left(\frac{1}{\delta}\right)}{n}+\log\left(\frac{|\mathcal{Y}|^2 n\theta^2}{\log|\mathcal{H}|}\right)\frac{\log|\mathcal{H}|}{n\theta^2}}\right)$$

*where $\theta > \frac{\log|\mathcal{H}|}{4|\mathcal{Y}|^2 n}$ is a margin parameter and $\theta_{f,\eta}(x,y)$ is a measure of the confidence of $H(\cdot)$ in its predictions. Specifically we have that*

$$\mathop{\mathbb{P}}_{(x,y)\in S}[\theta_{f,\eta}(x,y)\leq\theta] \leq \frac{|\mathcal{Y}|}{2}\prod_{c=1}^{C} Z_c \sqrt{\left(\frac{W_{c,+}+\frac{1}{n}}{W_{c,-}+\frac{1}{n}}\right)^{\theta}}.$$

*Proof sketch.* To prove the first part, we exploit that the weighted combination of triplet classifiers learned by TripletBoost can be approximated by a random combination of triplet classifiers based on their weights. The proof of the second part is similar to the proof of Theorem 2. □

At a first glance it seems that this bound does not depend on $m$, the number of available triplets. However, this dependence is implicit: $m$ impacts the probability that the training examples are well classified with a large margin $\theta$. If the number of triplets is small, the probability that the training examples are well classified with a given margin is small. This probability increases when the number of triplets increases.

To illustrate this behaviour, consider a fixed margin $\theta$. Assume that each of the $n$ training examples is classified by exactly one triplet classifier and that each triplet classifier abstains on all but one example. In this extreme case, the only way to have no error on the training set is to combine the $n$ triplet classifiers that do not abstain. For simplicity consider the case where all the triplet classifiers have uniform weight $\frac{1}{n}$ in the final classifier. Then the fixed margin will not be achieved when the number of training examples increases. The first term on the right hand side of the generalization bound will be 1, that is the bound predicts that the learned classifier might not generalize. This fact remains true for any weighting scheme. In this example, the best weighting scheme classifies with a margin at least $\theta$, at most $\frac{1}{\theta}$ training examples.

When the number of triplets increases, the value of $Z_c$ decreases, because the proportion of examples on which the selected triplet classifier abstains, $1-W_{c,+}-W_{c,-}$, decreases. Similarly, the proportion of training examples that are classified with a margin at least $\theta$ increases. Hence the first term on the right hand side of the generalization bound is greatly reduced and the learned classifier generalizes well.

To prove a bound that explicitly depends on $m$ would be of significant interest. However this is a very difficult problem, as it requires to use an explicit measure of complexity for general metric spaces, which is beyond the scope of this paper.

### 3.3 Lower bound on the number of triplets

In this section we investigate the minimum number of triplets that are necessary to ensure that our algorithm performs well. Ideally, we would like to obtain a lower bound on the number of triplets that are necessary to achieve a given accuracy. In this paper we take a first step in this direction by deriving a bound on the number of triplets that are necessary to ensure that the learned classifier does not abstain on any unseen example. Theorems 4 and 5 show that it abstains with high probability if it is learned using too few triplets or if it combines too few triplet classifiers.

**Theorem 4 (Lower bound on the probability that a strong classifier abstains).** *Let $n \geq 2$ be the number of training examples, $p = \frac{2n^k}{n^3}$ with $k \in \left[0, 3-\frac{\log(2)}{\log(n)}\right)$ be the probability*

*that a triplet is available in the triplet set $T$ and $C = \frac{n^\beta}{2}$ with $\beta \in \left[0, 1 + \frac{\log(n-1)}{\log(n)}\right]$ be the number of classifiers combined in the learned classifier. Let $\mathcal{A}$ be any algorithm learning a classifier $H(\cdot) = \arg\max_{y \in \mathcal{Y}} \left(\sum_{c=1}^{C} \alpha_c \mathbb{I}_{y \in h_c(\cdot)}\right)$ that combines several triplet classifiers using some weights $\alpha_c \in \mathbb{R}$. Assume that triplet classifiers that abstain on all the training examples have a weight of 0 (that is if $h_c(x_i) = \vartheta$ for all the examples $(x_i, y_i) \in S$ then $\alpha_c = 0$). Then the probability that $H$ abstains on a test example is bounded as follows:*

$$\mathop{\mathbb{P}}_{(x,y)\sim\mathcal{D}_S}[H(x) = \vartheta] \geq \left(1 - p + p\,(1-p)^n\right)^C. \tag{6}$$

*Proof sketch.* The proof of this theorem is based on the idea that a learned classifier $H$ abstains on a test example if each of the combined triplet classifiers either (i) abstains on every training example or (ii) abstains on the test example. Computing the probability that it happens gives Equation (6). □

To understand the implications of this theorem we consider a concrete example.

**Example 1.** *Assume that we build a linear combination of all possible triplet classifiers, that is $C = \frac{n(n-1)}{2}$. Then we have*

$$\lim_{n\to+\infty} \mathop{\mathbb{P}}_{(x,y)\sim\mathcal{D}_S}[H(x) = \vartheta] \geq \begin{cases} 1 & \text{if } k < \frac{3}{2}, \\ \exp(-2) & \text{if } k = \frac{3}{2}, \\ 0 & \text{if } \frac{3}{2} < k, \end{cases} \tag{7}$$

*where $k$ is the parameter that controls the probability $p$ that a particular triplet is available in the triplets set $T$. The bottom line is that when $k < \frac{3}{2}$, that is when we do not have at least $\Omega(n\sqrt{n})$ random triplets, the learned classifier abstains on all the examples.*

*Proof sketch.* This example follows from Theorem 4 by replacing $p$ and $C$ by their respective values and taking the limit of the right hand side of Equation 6 as $n$ goes to infinity. □

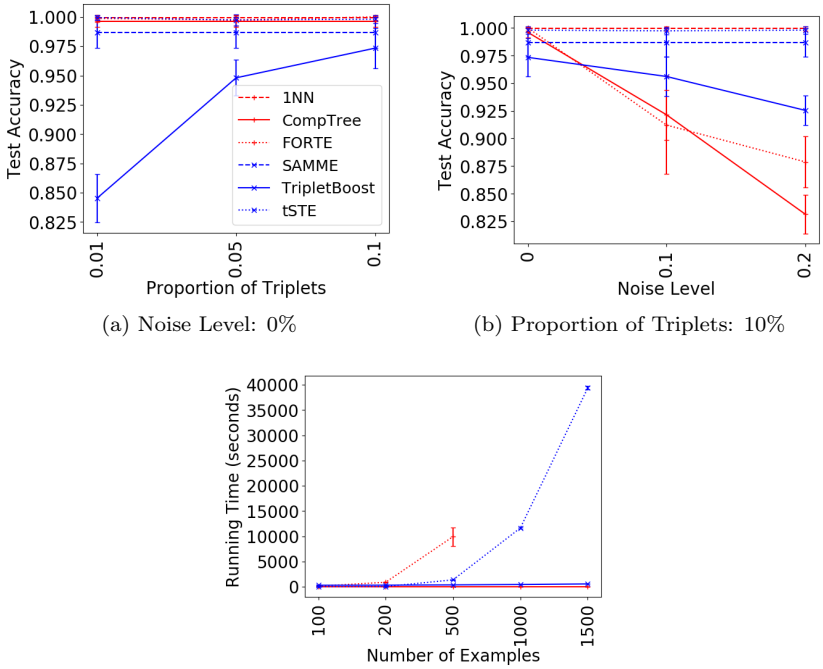
Theorem 4 shows that when $p$ and $C$ are too small, then the strong classifier abstains with high probability. However, the theorem does not guarantee that the strong classifier does not abstain when $p$ and $C$ are large. The next theorem takes care of this other direction, under slightly stronger assumptions on the weights learned by the algorithm.

**Theorem 5 (Exact bound on the probability that a strong classifier abstains).** *In Theorem 4, further assume that each triplet classifier that does not abstain on at least one training example has a weight different from 0 (if for at least one example $(x_i, y_i) \in S$ we have that $h_c(x_i) \neq \vartheta$ then $\alpha_c \neq 0$). Then equality holds in Equation (6).*

*Proof sketch.* The proof is very similar to the proof of Theorem 4. □

Theorem 5 implies that equality holds in Example 1, thus when $C = \frac{n(n-1)}{2}$ we need at least $k > \frac{3}{2}$, that is at least $\Omega(n\sqrt{n})$ random triplets, to obtain a classifier that never abstains. In the supplementary we present an extended version of Example 1 where we study the limit when $n \to \infty$ for general values of $C$ and $p$, and we provide a graphical illustration of the bound.

Example 1 shows that when $C = \frac{n(n-1)}{2}$ the number of available triplets should at least scale as $\Omega(n\sqrt{n})$ to obtain a non-trivial classifier that does not abstain on most of the test examples. The number of triplets that are necessary to achieve good classification accuracy might even be higher. Note that this lower bound does not contradict existing results (Ailon, 2012; Jamieson and Nowak, 2011; Jain et al., 2016). These results were developed in the different context of triplet recovery, where the goal is not classification, but to predict the outcome of unobserved triplet questions. For example it has been shown that to exactly recover all the triplets, the number of passively available triplets should scale in $\Omega(n^3)$ (Jamieson and Nowak, 2011). Similarly Jain et al. (2016) derive a finite error bound for approximate recovery of the Euclidean Gram matrix. Our bound simply tells us that, in a classification setting, it might be possible to do better than that. To set a complete picture, one would need to derive an upper bound on the number of triplets necessary for good accuracy in classification.




(a) Noise Level: 0%

(b) Proportion of Triplets: 10%

(c) Running Times, Proportion of Triplets: 10%, Noise Level: 10%

Figure 2: The Moons dataset is a small scale dataset with 500 examples and 2 dimensions. In Figure 2a we consider the noise free setting and we vary the proportion of triplets available from 1 to 10% of all the triplets. In Figure 2b we fix the proportion of available triplets to 10% of all the triplets and we vary the noise level from 0 to 20%. Figure 2c presents the running time of the training step of the triplet-based methods with training samples of increasing sizes. The proportion of triplets and the noise level were both set to 10%. The results were obtained on a single core @3.40GHz. For sizes bigger than 500 FORTE was stopped after 12 hours.

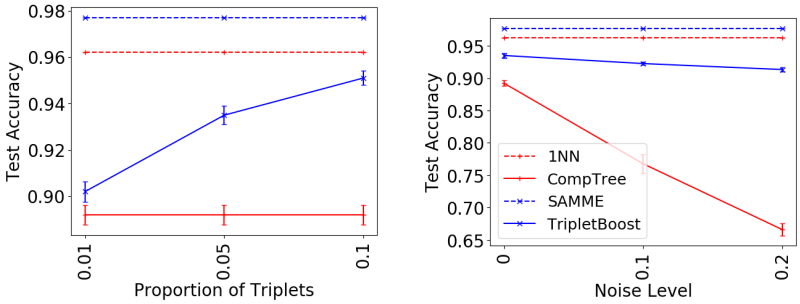
# 4 Experiments

In this section we propose an empirical evaluation of TripletBoost. We consider four datasets of varying scales and different baselines either based on ordinal embedding or able to directly deal with triplets.

**Baselines** First, we compare our approach to two ordinal embedding based approaches, tSTE (Van Der Maaten and Weinberger, 2012) and FORTE (Jain et al., 2016). The idea is to first embed the triplets in a Euclidean space and then to use the 1-nearest neighbour algorithm for classification. We also would like to compare to alternative approaches that are able to learn directly from triplets (without embedding as a first step). However, to the best of our knowledge, TripletBoost is the only method able to do classification using only passively obtained triplets. The only option is to choose competing methods that have access to more information than our own algorithm (providing an unfair advantage to the alternative methods). We settled for a method that uses actively chosen triplets to build a comparison-tree to retrieve nearest neighbours (CompTree) (Haghiri et al., 2017). Finally, to put the results obtained in the triplet setting in perspective, we consider two methods that use the original Euclidean representations of the data, the 1-nearest neighbour algorithm (1NN) and AdaBoost.SAMME (SAMME) (Hastie et al., 2009).

Table 1: Summary of the different datasets.

| NAME | DIM | TRAIN / TEST | CLASSES |
|---|---|---|---|
| Iris | 4 | 105 / 45 | 3 |
| Moons | 2 | 350 / 150 | 2 |
| Gisette | 5000 | 6000 / 1000 | 2 |
| CodRna | 8 | 59535 / 271617 | 2 |




(a) Noise Level: 0%

(b) Proportion of Triplets: 5%

Figure 3: The Gisette dataset has 7000 examples and 5000 dimensions. In Figure 3a we consider the noise free setting and we vary the proportion of triplets available from 1 to 10% of all the triplets. In Figure 3b we fix the proportion of available triplets to 5% of all the triplets and we vary the noise level from 0 to 20%.

**Implementation details** For tSTE and FORTE we used the implementations freely available from the authors' website. For both methods we set the embedding dimension to the original dimension of the data. Furthermore these two methods were only considered for small datasets with less than $10^3$ examples as they do not scale well to bigger datasets (as illustrated in Figure 2c). For CompTree we used our own implementation and the leaf size of the comparison tree has been set to 1 as this is the only value for which this method can handle noise. For 1NN and SAMME we used the sk-learn implementations (Pedregosa et al., 2011) and the number of boosting iterations for SAMME has been set to $10^3$. Finally for TripletBoost we set the number of boosting iterations to $10^6$.

**Datasets and performance measure** We consider four different datasets (see Table 1). For each dataset we use the Euclidean distance to generate some triplets as described in Equation (1). Given a set of $n$ examples there are $n^2(n-1)/2$ possible triplets. We consider three different regimes where 1%, 5% or 10% of the triplets are available, and we consider three noise levels where 0%, 10% or 20% of the triplets are incorrect. We end up with 9 different experimental settings for each dataset. We measure the performance of the different methods in terms of test accuracy (higher is better). For all the experiments we report the mean and standard deviation of 10 repetitions. Since the results are mostly consistent across the datasets we defer most of the plots to the supplementary material and we only present the most representative ones here.

**Results in the small scale regime** We first consider the small scale datasets where the number of training examples is lower than $10^3$ (Figure 2). In this setting our method does not perform well when the number of triplets is too small, but gets closer to the baseline methods when the number of triplets increases (Figure 2a). This behaviour can easily be explained: when only 1% of the triplets are available, the triplet classifiers abstain on all but 3 or 4 examples on average.

Hence, their performance evaluations are not reliable, and consequently the weights of the triplet classifiers cannot be chosen in a satisfactory manner. This problem vanishes when the number of triplets increases. Considering increasing noise levels (Figure 2b) one can notice that TripletBoost is more robust than CompTree. We believe that this is the case because the CompTree algorithm generates a comparison-tree based on individual triplet queries, and the greedy decisions in the tree building procedure can easily be misleading in the presence of noise. Our approach is much more robust in this regard.

**Results in the large scale regime** We also consider bigger datasets (Figure 3). In this case we can observe that our method does not quite reach the accuracy of the baseline methods 1NN and SAMME, who exploit a significant amount of extra information. Still, it performs quite well. Our method is competitive with CompTree, the method that uses active rather than passive queries. The ordinal embedding approaches cannot compete with our method in this regime, as they are too slow to even finish. Once again TripletBoost is quite resistant to noise (Figure 3b).

# 5 Conclusion

In this paper we proposed TripletBoost to address the problem of comparison-based classification. It is particularly designed for the situations where triplets cannot be queried actively, and we have to live with whatever set of triplets we get. We do not make any geometric assumptions on the underlying space. From a theoretical point of view we have shown that TripletBoost is well founded and we proved guarantees on both the training error and the generalization error of the learned classifier. Furthermore we derived a new lower bound showing that to avoid learning a random predictor, at least $\Omega(n\sqrt{n})$ triplets are needed. In practice we have shown that, given a sufficient amount of triplets, our method is competitive with state of the art methods but also that it is quite resistant to noise.

To the best of our knowledge, TripletBoost is the first algorithm that is able to handle large scale datasets using only passively obtained triplets. This means that the comparison-based setting could be considered for problems which were, until now, out of reach. As an illustration, consider a platform where users can watch, rate and comment movies. It is reasonable to assume that triplets of the form *movie $m_i$ is closer to movie $m_j$ than to movie $m_k$* can be automatically obtained using, for example, the ratings and the comments of the users or their interactions. In this scenario, active learning methods are not applicable since the users might be reluctant to answer solicitations. Similarly, embedding methods are too slow to handle large numbers of movies or users. However we can use TripletBoost to solve problems such as predicting the genres of the movies. As a proof of concept we considered the 1m movielens dataset (Harper and Konstan, 2016). It contains 1 million ratings from 6040 users on 3706 movies. We propose (i) to use the users' ratings to obtain some triplets about the movies, and (ii) to use TripletBoost to learn a classifier able to predict the genres of a new movie (more details on this experiment are given in the supplementary). Given a new movie, in $\sim$83% of the cases the genre predicted as the most likely one is correct and, on average, the 5 genres predicted as the most likely ones cover $\sim$92% of the true genres.

# References

Agarwal, S., Wills, J., Cayton, L., Lanckriet, G., Kriegman, D., and Belongie, S. (2007). Generalized non-metric multidimensional scaling. In *Artificial Intelligence and Statistics*, pages 11–18.

Ailon, N. (2012). An active learning algorithm for ranking from pairwise preferences with an almost optimal query complexity. *Journal of Machine Learning Research*, 13(Jan):137–164.

Amid, E. and Ukkonen, A. (2015). Multiview triplet embedding: Learning attributes in multiple maps. In *International Conference on Machine Learning*, pages 1472–1480.

Bellet, A., Habrard, A., and Sebban, M. (2015). Metric learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 9(1):1–151.

Breiman, L. (1999). Prediction games and arcing algorithms. *Neural computation*, 11(7):1493–1517.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139.

Gao, W. and Zhou, Z.-H. (2013). On the doubt about margin explanation of boosting. *Artificial Intelligence*, 203:1–18.

Haghiri, S., Ghoshdastidar, D., and von Luxburg, U. (2017). Comparison-based nearest neighbor search. In *Artificial Intelligence and Statistics*, pages 851–859.

Harper, F. M. and Konstan, J. A. (2016). The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):19.

Hastie, T., Rosset, S., Zhu, J., and Zou, H. (2009). Multi-class adaboost. *Statistics and its Interface*, 2(3):349–360.

Heikinheimo, H. and Ukkonen, A. (2013). The crowd-median algorithm. In *First AAAI Conference on Human Computation and Crowdsourcing*.

Jain, L., Jamieson, K. G., and Nowak, R. (2016). Finite sample prediction and recovery bounds for ordinal embedding. In *Advances In Neural Information Processing Systems*, pages 2711–2719.

Jamieson, K. G. and Nowak, R. D. (2011). Low-dimensional embedding using adaptively selected ordinal data. In *Conference on Communication, Control, and Computing*, pages 1077–1084.

Kane, D. M., Lovett, S., Moran, S., and Zhang, J. (2017). Active classification with comparison queries. In *Foundations of Computer Science (FOCS), 2017 IEEE 58th Annual Symposium on*, pages 355–366.

Kleindessner, M. and Luxburg, U. (2015). Dimensionality estimation without distances. In *Artificial Intelligence and Statistics*, pages 471–479.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Schapire, R. E. and Freund, Y. (2012). *Boosting: Foundations and algorithms*. MIT press.

Schapire, R. E., Freund, Y., Bartlett, P., and Lee, W. S. (1998). Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of statistics*, pages 1651–1686.

Schapire, R. E. and Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions. *Machine learning*, 37(3):297–336.

Schapire, R. E. and Singer, Y. (2000). Boostexter: A boosting-based system for text categorization. *Machine learning*, 39(2-3):135–168.

Tamuz, O., Liu, C., Belongie, S., Shamir, O., and Kalai, A. T. (2011). Adaptively learning the crowd kernel. In *International Conference on Machine Learning*, pages 673–680.

Terada, Y. and von Luxburg, U. (2014). Local ordinal embedding. In *International Conference on Machine Learning*, pages 847–855.

Van Der Maaten, L. and Weinberger, K. (2012). Stochastic triplet embedding. In *Machine Learning for Signal Processing (MLSP)*, pages 1–6.

Wang, L., Sugiyama, M., Jing, Z., Yang, C., Zhou, Z.-H., and Feng, J. (2011). A refined margin analysis for boosting algorithms via equilibrium margin. *Journal of Machine Learning Research*, 12(Jun):1835–1863.

# A Proof of Theorem 1

**Theorem 1 (Triplet classifiers and weak learners).** *Let $\mathcal{W}_c$ be an empirical distribution over $S \times \mathcal{Y}$ and $h_c$ be the corresponding triplet classifier chosen as described in Section 2.2. It holds that:*

1. *the error of $h_c$ on $\mathcal{W}_c$ is at most the error of a random predictor and is minimal compared to other labelling strategies,*
2. *the weight $\alpha_c$ of the classifier is non-zero if and only if, $h_c$ is a weak classifier, that is is strictly better than a random predictor.*

*Proof.* To prove the first claim of the theorem we first study the error of $h_c$ on $\mathcal{W}_c$ and then we show that any labelling strategy different from our would increase the proportion of incorrectly classified examples. To prove the second claim we simply use the definition of the weights $\alpha_c$.

**First claim** First of all notice that the probability that a triplet classifier $h_c$ makes an error on $\mathcal{W}_c$ is

$$\mathbb{P}_{(x_i,y)\sim\mathcal{W}_c} \left[ (\mathbb{I}_{y=y_i} - \mathbb{I}_{y\neq y_i}) \neq \left(\mathbb{I}_{y\in h_c(x_i)} - \mathbb{I}_{y\notin h_c(x_i)}\right) \right]$$

that is, by the law of total probabilities:

$$\begin{aligned} &\mathbb{P}_{\substack{(x_i,y)\sim\mathcal{W}_c \\ h_c(x_i)\neq\vartheta}} \left[ (\mathbb{I}_{y=y_i} - \mathbb{I}_{y\neq y_i}) \neq \left(\mathbb{I}_{y\in h_c(x_i)} - \mathbb{I}_{y\notin h_c(x_i)}\right) \right] \mathbb{P}_{(x_i,y)\sim\mathcal{W}_c} \left[ h_c(x_i) \neq \vartheta \right] \\ &\qquad + \mathbb{P}_{\substack{(x_i,y)\sim\mathcal{W}_c \\ h_c(x_i)=\vartheta}} \left[ (\mathbb{I}_{y=y_i} - \mathbb{I}_{y\neq y_i}) \neq \left(\mathbb{I}_{y\in h_c(x_i)} - \mathbb{I}_{y\notin h_c(x_i)}\right) \right] \mathbb{P}_{(x_i,y)\sim\mathcal{W}_c} \left[ h_c(x_i) = \vartheta \right]. \end{aligned}$$

Notice that when $h_c$ abstains the best possible behaviour is to randomly predict the labels and thus

$$\mathbb{P}_{\substack{(x_i,y)\sim\mathcal{W}_c \\ h_c(x_i)=\vartheta}} \left[ (\mathbb{I}_{y=y_i} - \mathbb{I}_{y\neq y_i}) \neq \left(\mathbb{I}_{y\in h_c(x_i)} - \mathbb{I}_{y\notin h_c(x_i)}\right) \right] = \frac{1}{2}.$$

Hence to show that $h_c$ has at most the error of a random predictor it is sufficient to show that

$$\mathbb{P}_{\substack{(x_i,y)\sim\mathcal{W}_c \\ h_c(x_i)\neq\vartheta}} \left[ (\mathbb{I}_{y=y_i} - \mathbb{I}_{y\neq y_i}) \neq \left(\mathbb{I}_{y\in h_c(x_i)} - \mathbb{I}_{y\notin h_c(x_i)}\right) \right] = \frac{W_{c,-}}{W_{c,+} + W_{c,-}} \leq \frac{1}{2}$$

where $W_{c,+}$ and $W_{c,-}$ are respectively the proportions of correctly and incorrectly classified examples.

On the one hand assume that $h_c$ uses $x_j$ and $x_k$ as reference points. Let $S_{c,j}$ be the set of all the training examples for which $h_c$ does not abstain and which are closer to $x_j$ than to $x_k$. If $y \in o_j$ then we have that:

$$\sum_{z_i\in S_{c,j}} \mathbb{I}_{y=y_i} w_{c,x_i,y} > \sum_{z_i\in S_{c,j}} \mathbb{I}_{y\neq y_i} w_{c,x_i,y}. \tag{8}$$

Similarly, if $y \notin o_j$ then it implies that:

$$\sum_{z_i \in S_{c,j}} \mathbb{I}_{y \neq y_i} w_{c,x_i,y} \geq \sum_{z_i \in S_{c,j}} \mathbb{I}_{y=y_i} w_{c,x_i,y}. \tag{9}$$

The same result holds for $o_k$.

On the other hand recall that:

$$\begin{aligned} W_{c,+} &= \sum_{\substack{z_i \in S \\ h_c(x_i) \neq \vartheta}} \left( \mathbb{I}_{y_i \in h_c(x_i)} w_{c,x_i,y_i} + \sum_{y \neq y_i} \mathbb{I}_{y \notin h_c(x_i)} w_{c,x_i,y} \right), \\ W_{c,-} &= \sum_{\substack{z_i \in S \\ h_c(x_i) \neq \vartheta}} \left( \mathbb{I}_{y_i \notin h_c(x_i)} w_{c,x_i,y_i} + \sum_{y \neq y_i} \mathbb{I}_{y \in h_c(x_i)} w_{c,x_i,y} \right). \end{aligned} \tag{10}$$

Combining Equations (8), (9), and (10) we obtain:

$$\begin{aligned} & W_{c,-} \leq W_{c,+} \\ \Leftrightarrow \quad & 2W_{c,-} \leq W_{c,+} + W_{c,-} \\ \Leftrightarrow \quad & \frac{W_{c,-}}{W_{c,+} + W_{c,-}} \leq \frac{1}{2} \end{aligned}$$

which proves that $h_c$ has at most the error of a random predictor.

To see that our labelling strategy is optimal just notice that any other labelling strategy would either predict some labels in $o_j$ or $o_k$ that do not conform with Equation (8) or not predict some labels from $o_j$ or $o_k$ that do not conform with Equation (9). As a consequence $W_{c,-}$ would be increased and thus the error of $h_c$ would also be increased. This concludes the proof of the first claim.

**Second claim** Recall that $\alpha_c$ is computed as

$$\alpha_c = \frac{1}{2} \log \left( \frac{W_{c,+} + \frac{1}{n}}{W_{c,-} + \frac{1}{n}} \right).$$

Also notice that $h_c$ is a weak classifier, that is better than a random predictor, when $W_{c,-} < W_{c,+}$. We have that:

$$\begin{aligned} & W_{c,-} < W_{c,+} \\ \Leftrightarrow \quad & W_{c,-} + \frac{1}{n} < W_{c,+} + \frac{1}{n} \\ \Leftrightarrow \quad & 1 < \frac{W_{c,+} + \frac{1}{n}}{W_{c,-} + \frac{1}{n}} \\ \Leftrightarrow \quad & 0 < \frac{1}{2} \log \left( \frac{W_{c,+} + \frac{1}{n}}{W_{c,-} + \frac{1}{n}} \right) \end{aligned}$$

and it proves the second claim. □

# B Proof of Theorem 2

**Theorem 2 (Reduction of the Training Error).** *Let $S$ be a set of $n$ examples and $T$ be a set of $m$ triplets (obtained as described in Section 2). Let $H(\cdot)$ be the classifier obtained after $C$ iterations of TripletBoost (Algorithm 1) using $S$ and $T$ as input. It holds that:*

$$\mathop{\mathbb{P}}_{(x,y) \in S} [H(x) \neq y] \leq \frac{|\mathcal{Y}|}{2} \prod_{c=1}^{C} Z_c$$

*with* $Z_c = (1 - W_{c,+} - W_{c,-}) + (W_{c,+}) \cdot \sqrt{\frac{W_{c,-}+\frac{1}{n}}{W_{c,+}+\frac{1}{n}}} + (W_{c,-}) \cdot \sqrt{\frac{W_{c,+}+\frac{1}{n}}{W_{c,-}+\frac{1}{n}}} \leq 1$.

*Proof sketch.* This result is inherited from AdaBoost.MO and can be obtained directly by applying Theorem 10.4 in Chapter 10 of the book of Schapire and Freund (2012) using a so-called loss-based decoding and noticing that in our case $\bar{K} = |\mathcal{Y}|$ and $\rho = 2$. □

# C Proof of Theorem 3

**Theorem 3 (Generalization Guarantees).** *Let $\mathcal{D}_{\mathcal{S}}$ be a distribution over $\mathcal{X} \times \mathcal{Y}$, let $S$ be a set of $n$ examples drawn i.i.d. from $\mathcal{D}_{\mathcal{S}}$, and let $T$ be a set of $m$ triplets (obtained as described in Section 2). Let $H(\cdot)$ be the classifier obtained after $C$ iterations of TripletBoost (Algorithm 1) using $S$ and $T$ as input. Let $\mathcal{H}$ be a set of triplet classifiers as defined in Section 2.1. Then, with probability at least $1 - \delta$, we have*

$$\mathop{\mathbb{P}}_{(x,y)\sim\mathcal{D}_{\mathcal{S}}}[H(x) \neq y] \leq \mathop{\mathbb{P}}_{(x,y)\in S}[\theta_{f,\eta}(x,y) \leq \theta] + \mathcal{O}\left(\sqrt{\frac{\log\left(\frac{1}{\delta}\right)}{n} + \log\left(\frac{|\mathcal{Y}|^2 n\theta^2}{\log|\mathcal{H}|}\right)\frac{\log|\mathcal{H}|}{n\theta^2}}\right)$$

*where $\theta > \frac{\log|\mathcal{H}|}{4|\mathcal{Y}|^2 n}$ is a margin parameter and $\theta_{f,\eta}(x,y)$ is a measure of the confidence of $H(\cdot)$ in its predictions. Specifically we have that*

$$\mathop{\mathbb{P}}_{(x,y)\in S}[\theta_{f,\eta}(x,y) \leq \theta] \leq \frac{|\mathcal{Y}|}{2}\prod_{c=1}^{C} Z_c \sqrt{\left(\frac{W_{c,+}+\frac{1}{n}}{W_{c,-}+\frac{1}{n}}\right)^{\theta}}.$$

*Proof sketch.* This result is inherited from AdaBoost.MO and can be obtained directly by following the steps of Exercise 10.3 in Chapter 10 of the book of Schapire and Freund (2012). Following our notations $\theta_{f,\eta}(x,y)$ is equal to $\mathcal{M}_{f,\eta}(x,y)$ and the free parameter $n$ in their proof has been chosen as $\left\lceil \frac{16}{\theta^2}\log\left(\frac{4|\mathcal{Y}|^2 n\theta^2}{\log|\mathcal{H}|}\right)\right\rceil$. The second part of the theorem can be obtained in a similar way by following the steps of Exercise 10.4 in Chapter 10 of the book of Schapire and Freund (2012). □

# D Proof of Theorem 4

**Theorem 4 (Lower bound on the probability that a strong classifier abstains).** *Let $n \geq 2$ be the number of training examples, $p = \frac{2n^k}{n^3}$ with $k \in \left[0, 3 - \frac{\log(2)}{\log(n)}\right)$ be the probability that a triplet is available in the triplet set $T$ and $C = \frac{n^\beta}{2}$ with $\beta \in \left[0, 1 + \frac{\log(n-1)}{\log(n)}\right]$ be the number of classifiers combined in the learned classifier. Let $\mathcal{A}$ be any algorithm learning a classifier $H(\cdot) = \arg\max_{y\in\mathcal{Y}}\left(\sum_{c=1}^{C}\alpha_c \mathbb{I}_{y\in h_c(\cdot)}\right)$ that combines several triplet classifiers using some weights $\alpha_c \in \mathbb{R}$. Assume that triplet classifiers that abstain on all the training examples have a weight of 0 (that is if $h_c(x_i) = \vartheta$ for all the examples $(x_i, y_i) \in S$ then $\alpha_c = 0$). Then the probability that $H$ abstains on a test example is bounded as follows:*

$$\mathop{\mathbb{P}}_{(x,y)\sim\mathcal{D}_{\mathcal{S}}}[H(x) = \vartheta] \geq \left(1 - p + p\,(1-p)^n\right)^C. \tag{11}$$

*Proof.* Recall that $H(\cdot) = \arg\max_{y\in\mathcal{Y}}\left(\sum_{c=1}^{C}\alpha_c \mathbb{I}_{y\in h_c(\cdot)}\right)$. Each component $\alpha_c \mathbb{I}_{y\in h_c(\cdot)}$ of $H$ abstains for a given example $x$ if $h_c(x) = \vartheta$ or $\alpha_c = 0$. Overall $H$ abstains if all its components also abstain and we bound this probability here.

Recall that $S = \{(x_i, y_i)\}_{i=1}^n$ is a set of $n$ examples drawn i.i.d. from an unknown distribution $\mathcal{D}_\mathcal{S}$ over the space $\mathcal{X} \times \mathcal{Y}$ where $|\mathcal{Y}| < \infty$ and recall that $C$ is the number of classifiers selected in $H$. Finally recall that each triplet is obtained with probability $p$ independently of the other triplets. It implies that each triplet classifier $h_c(\cdot)$ abstains with probability $q = 1 - p$ independently of the other triplet classifiers.

First we start by defining several random variables. Let $Y_{1,1}, \ldots, Y_{n,C}$ be $nC$ independent random variables encoding the event that a classifier abstains on one example:

$$Y_{i,c} = \begin{cases} 0 & \text{if } h_c \text{ abstains for example } x_i, \\ 1 & \text{otherwise.} \end{cases} \tag{12}$$

From the definition of the classifiers each of these random variables follow a Bernoulli distribution $\mathrm{B}(1, p)$.

Let $Y_{x,1}, \ldots, Y_{x,C}$ be $C$ independent random variables encoding the event that a classifier abstains on a new example $x$:

$$Y_{x,c} = \begin{cases} 0 & \text{if } h_c \text{ abstains for example } x, \\ 1 & \text{otherwise.} \end{cases} \tag{13}$$

From the definition of the classifiers each of these random variables follow a Bernoulli distribution $\mathrm{B}(1, p)$.

Let $V_1, \ldots, V_C$ be $C$ independent random variables encoding the probability that a triplet classifier abstains on all the training examples:

$$V_c = \sum_{i=1}^{n} Y_{i,c}. \tag{14}$$

As a sum of $n$ independent Bernoulli trials, these random variables follow a Binomial distribution $\mathrm{B}(n, p)$.

Using the law of total probabilities we have that:

$$\begin{aligned} \mathop{\mathbb{P}}_{(x,y)\sim\mathcal{D}_\mathcal{S}}[H(x) = \vartheta] = & \mathop{\mathbb{P}}_{(x,y)\sim\mathcal{D}_\mathcal{S}}\left[H(x) = \vartheta \,\middle|\, \sum_{c=1}^{C} V_c Y_{x,c} = 0\right] \mathop{\mathbb{P}}_{(x,y)\sim\mathcal{D}_\mathcal{S}}\left[\sum_{c=1}^{C} V_c Y_{x,c} = 0\right] \\ & + \mathop{\mathbb{P}}_{(x,y)\sim\mathcal{D}_\mathcal{S}}\left[H(x) = \vartheta \,\middle|\, \sum_{c=1}^{C} V_c Y_{x,c} \neq 0\right] \mathop{\mathbb{P}}_{(x,y)\sim\mathcal{D}_\mathcal{S}}\left[\sum_{c=1}^{C} V_c Y_{x,c} \neq 0\right]. \end{aligned}$$

Note that from the assumption that $\forall (x_i, y_i) \in S, h_c(x_i) = \vartheta$ then $\alpha_c = 0$ and the definition of the random variables $Y_{\mathbf{x},\cdot}$ we have that:

$$\mathop{\mathbb{P}}_{(x,y)\sim\mathcal{D}_\mathcal{S}}\left[H(x) = \vartheta \,\middle|\, \sum_{c=1}^{C} V_c Y_{x,c} = 0\right] = 1.$$

Similarly, we have that:

$$\mathop{\mathbb{P}}_{(x,y)\sim\mathcal{D}_\mathcal{S}}\left[H(x) = \vartheta \,\middle|\, \sum_{c=1}^{C} V_c Y_{x,c} \neq 0\right] \geq 0. \tag{15}$$

Hence we have that:

$$\mathop{\mathbb{P}}_{(x,y)\sim\mathcal{D}_\mathcal{S}}[H(x) = \vartheta] \geq \mathop{\mathbb{P}}_{(x,y)\sim\mathcal{D}_\mathcal{S}}\left[\sum_{c=1}^{C} V_c Y_{x,c} = 0\right]. \tag{16}$$

Let $U_1, \ldots, U_C$ be $C$ independent random variables such that:

$$U_c = \begin{cases} 1 & \text{if } V_c Y_{\mathbf{x},c} = 0, \\ 0 & \text{otherwise.} \end{cases}. \tag{17}$$

These random variables follow a Bernoulli distribution $\mathrm{B}\left(1,(1-p)+p(1-p)^n\right)$. To see that, note that $V_c$ and $Y_{x,c}$ are independent and thus we have that:

$$\begin{aligned}
\mathop{\mathbb{P}}_{(x,y)\sim\mathcal{D}_S}\left[V_cY_{x,c}=0\right] &= 1-\mathop{\mathbb{P}}_{(x,y)\sim\mathcal{D}_S}\left[V_cY_{x,c}\neq 0\right] \\
&= 1-\mathop{\mathbb{P}}_{(x,y)\sim\mathcal{D}_S}\left[\{V_c\neq 0\}\cap\{Y_{x,c}\neq 0\}\right] \\
&= 1-\mathop{\mathbb{P}}_{(x,y)\sim\mathcal{D}_S}\left[V_c\neq 0\right]\mathop{\mathbb{P}}_{(x,y)\sim\mathcal{D}_S}\left[Y_{x,c}\neq 0\right] \\
&\qquad\qquad (Y_{x,c} \text{ follows a Bernoulli distribution } \mathrm{B}\left(1,p\right).) \\
&= 1-\left(1-\mathop{\mathbb{P}}_{(x,y)\sim\mathcal{D}_S}\left[V_c=0\right]\right)p \\
&\qquad\qquad (V_c \text{ follows a Binomial distribution } \mathrm{B}\left(n,p\right).) \\
&= 1-\left(1-\binom{n}{0}p^0\left(1-p\right)^{n-0}\right)p \\
&= 1-\left(1-\left(1-p\right)^n\right)p \\
&= \left(1-p\right)+p\left(1-p\right)^n.
\end{aligned}$$

The r.h.s. of Inequality (16) can then be written as:

$$\mathop{\mathbb{P}}_{(x,y)\sim\mathcal{D}_S}\left[\sum_{c=1}^{C}V_cY_{x,c}=0\right]=\mathop{\mathbb{P}}_{(x,y)\sim\mathcal{D}_S}\left[\sum_{c=1}^{C}U_c=c\right] \tag{18}$$

which, by definition of the random variables $U_\cdot$, corresponds to the probability of obtaining $c$ successes among $c$ Bernoulli trials. In other words the random variable $U=\sum_{c=1}^{C}U_j$ follows a Binomial distribution $\mathrm{B}\left(c,(1-p)+p(1-p)^n\right)$. Following this we have that:

$$\begin{aligned}
\mathop{\mathbb{P}}_{(x,y)\sim\mathcal{D}_S}\left[\sum_{c=1}^{C}V_cY_{x,c}=0\right] &= \binom{c}{c}\left(\left(1-p\right)+p(1-p)^n\right)^C\left(1-\left(1-p\right)+p(1-p)^n\right)^{C-C} &(19)\\
&= \left(\left(1-p\right)+p(1-p)^n\right)^C. &(20)
\end{aligned}$$

Hence plugging this result in Inequality 16 we have that:

$$\mathop{\mathbb{P}}_{(x,y)\sim\mathcal{D}_S}\left[H(x)=\vartheta\right]\geq\left(\left(1-p\right)+p(1-p)^n\right)^C$$

which concludes the proof. □

# E Proof of Example 1

Note that here we prove a more general version of Example 4 than the one presented in the main paper. The latter follows directly by choosing $\beta=1+\frac{\log\left(n-1\right)}{\log\left(n\right)}$.

**Example 1.** *In Theorem 4, we choose different values for $p=2n^{k-3}$ the probability of that a triplet is available and $C=\frac{n^\beta}{2}$ the number of combined classifiers. Then we take the limit as $n\to\infty$ to obtain the following results.*

*If $0\leq\beta<1$ then:*

$$\lim_{n\to+\infty}\mathop{\mathbb{P}}_{(x,y)\sim\mathcal{D}_S}\left[H(x)=\vartheta\right]\geq\begin{cases}1 & \text{if } 0\leq k<3-\beta,\\ \exp(-1) & \text{if } k=3-\beta,\\ 0 & \text{if } 3-\beta<k<3-\frac{\log(2)}{\log(n)}.\end{cases}$$

*If $\beta = 1$ then:*

$$\lim_{n\to+\infty} \mathop{\mathbb{P}}_{(x,y)\sim\mathcal{D}_{\mathcal{S}}} [H(x) = \vartheta] \geq \begin{cases} 1 & \text{if } 0 \leq k < 2, \\ \exp(\exp(-2) - 1) & \text{if } k = 2, \\ 0 & \text{if } 2 < k < 3 - \frac{\log(2)}{\log(n)}. \end{cases}$$

*If $1 < \beta \leq 1 + \frac{\log(n-1)}{\log(n)}$ then:*

$$\lim_{n\to+\infty} \mathop{\mathbb{P}}_{(x,y)\sim\mathcal{D}_{\mathcal{S}}} [H(x) = \vartheta] \geq \begin{cases} 1 & \text{if } 0 \leq k < \frac{5-\beta}{2}, \\ \exp(-2) & \text{if } k = \frac{5-\beta}{2}, \\ 0 & \text{if } \frac{5-\beta}{2} < k < 3 - \frac{\log(2)}{\log(n)}. \end{cases} \tag{21}$$

*Proof.* Replacing $p$ and $C$ by their values in Equation (6) of Theorem 4 and applying Lemma 1 gives the example. □

The next Lemma is a technical result used in the proof of Theorem 4.

**Lemma 1 (Limit of the right hand side of Theorem 4.).** *Given $n \geq 2$, $k \in \left[0, 3 - \frac{\log(2)}{\log(n)}\right)$ and $\beta \in \left[0, 1 + \frac{\log(n-1)}{\log(n)}\right]$, define:*

$$f(n, k, \beta) \doteq \left(1 - 2n^{k-3} + 2n^{k-3}\left(1 - 2n^{k-3}\right)^n\right)^{\frac{n^\beta}{2}} \tag{22}$$

*then the following limits hold:*
*If $0 \leq \beta < 1$ then:*

$$\lim_{n\to+\infty} f(n, k, \beta) = \begin{cases} 1 & \text{if } 0 \leq k < 3 - \beta, \\ \exp(-1) & \text{if } k = 3 - \beta, \\ 0 & \text{if } 3 - \beta < k < 3 - \frac{\log(2)}{\log(n)}. \end{cases}$$

*If $\beta = 1$ then:*

$$\lim_{n\to+\infty} f(n, k, \beta) = \begin{cases} 1 & \text{if } 0 \leq k < 2, \\ \exp(\exp(-2) - 1) & \text{if } k = 2, \\ 0 & \text{if } 2 < k < 3 - \frac{\log(2)}{\log(n)}. \end{cases}$$

*If $1 < \beta \leq 1 + \frac{\log(n-1)}{\log(n)}$ then:*

$$\lim_{n\to+\infty} f(n, k, \beta) = \begin{cases} 1 & \text{if } 0 \leq k < \frac{5-\beta}{2}, \\ \exp(-2) & \text{if } k = \frac{5-\beta}{2}, \\ 0 & \text{if } \frac{5-\beta}{2} < k < 3 - \frac{\log(2)}{\log(n)}. \end{cases}$$

*Proof.* We are looking for the limit when $n \to \infty$ of the function defined in Equation (22).

$$\begin{aligned} &\lim_{n\to+\infty} \left(1 - 2n^{k-3} + 2n^{k-3}\left(1 - 2n^{k-3}\right)^n\right)^{\frac{n^\beta}{2}} \\ &= \lim_{n\to+\infty} \exp\left(\frac{n^\beta}{2} \log\left(1 - 2n^{k-3} + 2n^{k-3}\left(1 - 2n^{k-3}\right)^n\right)\right) \\ &\qquad\qquad \text{(Maclaurin series of log since } \left|2n^{k-3} - 2n^{k-3}\left(1 - 2n^{k-3}\right)^n\right| < 1.\text{)} \\ &= \lim_{n\to+\infty} \exp\left(\frac{n^\beta}{2}\left(-\sum_{l=1}^{\infty} \frac{\left(2n^{k-3} - 2n^{k-3}\left(1 - 2n^{k-3}\right)^n\right)^l}{l}\right)\right) \end{aligned}$$

$$= \lim_{n\to+\infty} \exp\left(-\sum_{l=1}^{\infty} \frac{n^\beta}{2l}\left(2n^{k-3} - 2n^{k-3}\left(1-2n^{k-3}\right)^n\right)^l\right)$$

$$= \lim_{n\to+\infty} \exp\left(\underbrace{-\frac{n^\beta}{2}\left(2n^{k-3} - 2n^{k-3}\left(1-2n^{k-3}\right)^n\right)}_{\text{Main term, with } l=1}\right.$$

$$\left.\underbrace{-\sum_{l=2}^{\infty} \frac{n^\beta}{2l}\left(2n^{k-3} - 2n^{k-3}\left(1-2n^{k-3}\right)^n\right)^l}_{\text{Remaining term, with } l>1}\right). \quad (23)$$

In the following we study the limits of the two underlined terms in Equation (23).

**Main term, with $l = 1$.** To obtain the limit of this term we proceed in two steps. First we decompose it in a product of two sub terms and we study their individual limits. Then we consider the indeterminate forms that arise after considering the products of the limits and we obtain the correct limits by upper and lower bounding the term and showing that the limits of the two bounds are identical.

We are interested in:

$$\lim_{n\to+\infty} -\frac{n^\beta}{2}\left(2n^{k-3} - 2n^{k-3}\left(1-2n^{k-3}\right)^n\right) = \lim_{n\to+\infty} -n^{k-3+\beta}\left(1-\left(1-2n^{k-3}\right)^n\right).$$

- On the one hand, since $\beta \in \left[0, 1+\frac{\log(n-1)}{\log(n)}\right]$ we have:

$$\lim_{n\to+\infty} -n^{k-3+\beta} = \begin{cases} 0 & \text{if } 0 \le k < 3-\beta, \\ -1 & \text{if } k = 3-\beta, \\ -\infty & \text{if } 3-\beta < k < 3-\frac{\log(2)}{\log(n)}. \end{cases} \quad (24)$$

- On the other hand, we have:

$$\lim_{n\to+\infty} \left(1-\left(1-2n^{k-3}\right)^n\right) = \begin{cases} 0 & \text{if } 0 \le k < 2, \\ 1-\exp(-2) & \text{if } k = 2, \\ 1 & \text{if } 2 < k < 3-\frac{\log(2)}{\log(n)}. \end{cases} \quad (25)$$

To see that note that:

$$\lim_{n\to+\infty} \left(1-2n^{k-3}\right)^n = \exp\left(\lim_{n\to+\infty} n\log\left(1-2n^{k-3}\right)\right)$$

which is an indeterminate form that can be solved using l'Hôpital's rule. Given the derivatives:

$$\frac{\partial \log\left(1-2n^{k-3}\right)}{\partial n} = \frac{-2(k-3)n^{k-4}}{1-2n^{k-3}},$$
$$\frac{\partial \frac{1}{n}}{\partial n} = \frac{-1}{n^2},$$

it follows that:

$$\lim_{n\to+\infty} n\log\left(1-2n^{k-3}\right) = \lim_{n\to+\infty} -\frac{2(3-k)n^{k-2}}{1-2n^{k-3}}$$

$$= \begin{cases} 0 & \text{if } 0 \leq k < 2, \\ -2 & \text{if } k = 2, \\ -\infty & \text{if } 2 < k < 3 - \frac{\log(2)}{\log(n)}. \end{cases}$$

It implies:

$$\lim_{n \to +\infty} \left(1 - 2n^{k-3}\right)^n = \begin{cases} 1 & \text{if } 0 \leq k < 2, \\ \exp(-2) & \text{if } k = 2, \\ 0 & \text{if } 2 < k < 3 - \frac{\log(2)}{\log(n)}. \end{cases}$$

Combining Limits (24) and (25) we obtain the following limits:
If $0 \leq \beta < 1$ then:

$$\lim_{n \to +\infty} -n^{k-3+\beta} \left(1 - \left(1 - 2n^{k-3}\right)^n\right) = \begin{cases} 0 & \text{if } 0 \leq k < 2, \\ 0 & \text{if } k = 2, \\ 0 & \text{if } 2 < k < 3 - \beta, \\ -1 & \text{if } k = 3 - \beta, \\ -\infty & \text{if } 3 - \beta < k < 3 - \frac{\log(2)}{\log(n)}. \end{cases}$$

If $\beta = 1$ then:

$$\lim_{n \to +\infty} -n^{k-3+\beta} \left(1 - \left(1 - 2n^{k-3}\right)^n\right) = \begin{cases} 0 & \text{if } 0 \leq k < 2, \\ \exp(-2) - 1 & \text{if } k = 2, \\ -\infty & \text{if } 2 < k < 3 - \frac{\log(2)}{\log(n)}. \end{cases}$$

If $1 < \beta \leq 2$ then:

$$\lim_{n \to +\infty} -n^{k-3+\beta} \left(1 - \left(1 - 2n^{k-3}\right)^n\right) = \begin{cases} 0 & \text{if } 0 \leq k < 3 - \beta, \\ 0 & \text{if } k = 3 - \beta, \\ \text{Indeterminate} & \text{if } 3 - \beta < k < 2, \\ -\infty & \text{if } k = 2, \\ -\infty & \text{if } 2 < k < 3 - \frac{\log(2)}{\log(n)}. \end{cases} \tag{26}$$

The only indeterminate form arises when $1 < \beta \leq 2$ and $3 - \beta < k < 2$. To solve this indeterminate form we propose to upper and lower bound $-n^{k-3+\beta}\left(1 - \left(1 - 2n^{k-3}\right)^n\right)$ and to show that the limits coincides. Notice that, given our assumptions on $k$ we have that $-2n^{k-3} \in (-1, 0]$ and hence, since $n \geq 2$ we can apply Bernoulli's inequalities[2] to obtain:

$$\begin{aligned} & \left(1 - 2n^{k-2}\right) \leq \left(1 - 2n^{k-3}\right)^n \leq \left(1 - \frac{2n^{k-2}}{1 + 2n^{k-2} - 2n^{k-3}}\right) \\ \Leftrightarrow \quad & 1 - \left(1 - 2n^{k-2}\right) \geq 1 - \left(1 - 2n^{k-3}\right)^n \geq 1 - \left(1 - \frac{2n^{k-2}}{1 + 2n^{k-2} - 2n^{k-3}}\right) \\ \Leftrightarrow \quad & 2n^{k-2} \geq 1 - \left(1 - 2n^{k-3}\right)^n \geq \frac{2n^{k-2}}{1 + 2n^{k-2} - 2n^{k-3}} \\ \Leftrightarrow \quad & -2n^{2k-5+\beta} \leq -n^{k-3+\beta}\left(1 - \left(1 - 2n^{k-3}\right)^n\right) \leq -\frac{2n^{2k-5+\beta}}{1 + 2n^{k-2} - 2n^{k-3}} \end{aligned}$$

Furthermore, recalling that we are only interested in the case where $1 < \beta \leq 2$ and $3 - \beta < k < 2$, we have the following limits:

$$\lim_{n \to +\infty} -2n^{2k-5+\beta} = \begin{cases} 0 & \text{if } 3 - \beta < k < \frac{5-\beta}{2}, \\ -2 & \text{if } k = \frac{5-\beta}{2}, \\ -\infty & \text{if } \frac{5-\beta}{2} < k < 2. \end{cases}$$

[2] $1 + \frac{nx}{1-nx+x} \geq (1+x)^n \geq 1 + nx$ for $x \in (-1, 0]$ and $n \in \mathbb{N}$. Note that these inequalities might hold for more general values of $x$ and $n$ but we restricted ourselves to the case of interest for the proof.

$$\lim_{n\to+\infty} -\frac{2n^{2k-5+\beta}}{1+2n^{k-2}-2n^{k-3}} = \begin{cases} 0 & \text{if } 3-\beta < k < \frac{5-\beta}{2}, \\ -2 & \text{if } k = \frac{5-\beta}{2}, \\ -\infty & \text{if } \frac{5-\beta}{2} < k < 2. \end{cases}$$

Hence we obtain:

$$\lim_{n\to+\infty} -n^{k-3+\beta}\left(1-\left(1-2n^{k-3}\right)^n\right) = \begin{cases} 0 & \text{if } 3-\beta < k < \frac{5-\beta}{2}, \\ -2 & \text{if } k = \frac{5-\beta}{2}, \\ -\infty & \text{if } \frac{5-\beta}{2} < k < 2. \end{cases}$$

Combining this result with Equation (26) gives the limit of the main term:

If $0 \leq \beta < 1$ then:

$$\lim_{n\to+\infty} -n^{k-3+\beta}\left(1-\left(1-2n^{k-3}\right)^n\right) = \begin{cases} 0 & \text{if } 0 \leq k < 3-\beta, \\ -1 & \text{if } k = 3-\beta, \\ -\infty & \text{if } 3-\beta < k < 3-\frac{\log(2)}{\log(n)}. \end{cases}$$

If $\beta = 1$ then:

$$\lim_{n\to+\infty} -n^{k-3+\beta}\left(1-\left(1-2n^{k-3}\right)^n\right) = \begin{cases} 0 & \text{if } 0 \leq k < 2, \\ \exp(-2)-1 & \text{if } k = 2, \\ -\infty & \text{if } 2 < k < 3-\frac{\log(2)}{\log(n)}. \end{cases}$$

If $1 < \beta \leq 2$ then:

$$\lim_{n\to+\infty} -n^{k-3+\beta}\left(1-\left(1-2n^{k-3}\right)^n\right) = \begin{cases} 0 & \text{if } 0 \leq k < \frac{5-\beta}{2}, \\ -2 & \text{if } k = \frac{5-\beta}{2}, \\ -\infty & \text{if } \frac{5-\beta}{2} < k < 3-\frac{\log(2)}{\log(n)}. \end{cases} \tag{27}$$

**Remaining term, with $l > 1$.** We can rewrite the remaining term as follows:

$$\begin{aligned} -\sum_{l=2}^{\infty} \frac{n^\beta}{2l}\left(2n^{k-3}-2n^{k-3}\left(1-2n^{k-3}\right)^n\right)^l &= -\sum_{l=2}^{\infty} \frac{n^\beta}{2l} 2n^{kl-3l}\left(1-\left(1-2n^{k-3}\right)^n\right)^l \\ &= -\sum_{l=2}^{\infty} \frac{\left(1-\left(1-2n^{k-3}\right)^n\right)^l}{l} n^{(k-3)l+\beta} \end{aligned}$$

Notice that $\forall k \in \left[0, 3-\frac{\log(2)}{\log(n)}\right), \forall l \geq 2, \forall n \geq 2$ we have that $\left(1-\left(1-2n^{k-3}\right)^n\right)^l \in [0,1]$. Following this we have:

$$\begin{aligned} l > \frac{\beta}{3-k} &\Rightarrow \lim_{n\to+\infty} -n^{(k-3)l+\beta} = 0 \Rightarrow \lim_{n\to+\infty} -\frac{\left(1-\left(1-2n^{k-3}\right)^n\right)^l}{l} n^{(k-3)l+\beta} = 0 \\ l \leq \frac{\beta}{3-k} &\Rightarrow \lim_{n\to+\infty} -n^{(k-3)l+\beta} \leq 0 \Rightarrow \lim_{n\to+\infty} -\frac{\left(1-\left(1-2n^{k-3}\right)^n\right)^l}{l} n^{(k-3)l+\beta} \leq 0. \end{aligned}$$

From this, noticing that $k < \frac{6-\beta}{2}$ implies $\frac{\beta}{3-k} < 2$, we obtain:

$$0 \leq k < \frac{6-\beta}{2} \Rightarrow \lim_{n\to+\infty} -\sum_{l=2}^{\infty} \frac{n^2}{2l}\left(2n^{k-3}-2n^{k-3}\left(1-2n^{k-3}\right)^n\right)^l = 0 \tag{28}$$

$$\frac{6-\beta}{2} \leq k < 3-\frac{\log(2)}{\log(n)} \Rightarrow \lim_{n\to+\infty} -\sum_{l=2}^{\infty} \frac{n^2}{2l}\left(2n^{k-3}-2n^{k-3}\left(1-2n^{k-3}\right)^n\right)^l \leq 0 \tag{29}$$

**Limit of Equation** (23)**.** We now combine the limits obtained in Equations (27), (28) and (29) to obtain the following limits:

If $0 \leq \beta < 1$ then $\frac{6-\beta}{2} > 3 - \beta$ and thus the remaining term does not change the limit of the main term which implies:

$$\lim_{n \to +\infty} -\frac{n^\beta}{2}\left(2n^{k-3} - 2n^{k-3}\left(1 - 2n^{k-3}\right)^n\right) - \sum_{l=2}^{\infty} \frac{n^\beta}{2l}\left(2n^{k-3} - 2n^{k-3}\left(1 - 2n^{k-3}\right)^n\right)^l = \begin{cases} 0 & \text{if } 0 \leq k < 3 - \beta, \\ -1 & \text{if } k = 3 - \beta, \\ -\infty & \text{if } 3 - \beta < k < 3 - \frac{\log(2)}{\log(n)}. \end{cases}$$

If $\beta = 1$ then $\frac{6-\beta}{2} > 2$ and thus the remaining term does not change the limit of the main term which implies:

$$\lim_{n \to +\infty} -\frac{n^\beta}{2}\left(2n^{k-3} - 2n^{k-3}\left(1 - 2n^{k-3}\right)^n\right) - \sum_{l=2}^{\infty} \frac{n^\beta}{2l}\left(2n^{k-3} - 2n^{k-3}\left(1 - 2n^{k-3}\right)^n\right)^l = \begin{cases} 0 & \text{if } 0 \leq k < 2, \\ \exp(-2) - 1 & \text{if } k = 2, \\ -\infty & \text{if } 2 < k < 3 - \frac{\log(2)}{\log(n)}. \end{cases}$$

If $1 < \beta \leq 1 + \frac{\log(n-1)}{\log(n)}$ then $\frac{6-\beta}{2} > \frac{5-\beta}{2}$ and thus the remaining term does not change the limit of the main term which implies:

$$\lim_{n \to +\infty} -\frac{n^\beta}{2}\left(2n^{k-3} - 2n^{k-3}\left(1 - 2n^{k-3}\right)^n\right) - \sum_{l=2}^{\infty} \frac{n^\beta}{2l}\left(2n^{k-3} - 2n^{k-3}\left(1 - 2n^{k-3}\right)^n\right)^l = \begin{cases} 0 & \text{if } 0 \leq k < \frac{5-\beta}{2}, \\ -2 & \text{if } k = \frac{5-\beta}{2}, \\ -\infty & \text{if } \frac{5-\beta}{2} < k < 3 - \frac{\log(2)}{\log(n)}. \end{cases}$$

Plugging this result back into Equation (23) gives the lemma. □

# F Proof of Theorem 5

**Theorem 5 (Exact bound on the probability that a strong classifier abstains).** *In Theorem 4, further assume that each triplet classifier that does not abstain on at least one training example has a weight different from* 0 *(if for at least one example* $(x_i, y_i) \in S$ *we have that* $h_c(x_i) \neq \vartheta$ *then* $\alpha_c \neq 0$*). Then equality holds in Equation* (6)*.*

*Proof.* The theorem follows directly by noticing that, in the proof of Theorem 4, if we further assume that each triplet classifier that does not abstain on at least one training example has a weight different from 0 (if for at least one example $(x_i, y_i) \in S$ we have that $h_c(x_i) \neq \vartheta$ then $\alpha_c \neq 0$) then we have equality in Equation (15):

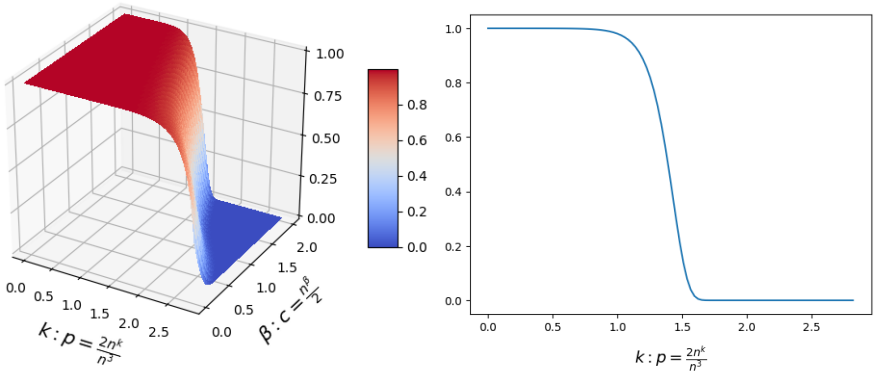
$$\mathop{\mathbb{P}}_{(x,y)\sim\mathcal{D}_S}\left[H(x) = \vartheta \,\middle|\, \sum_{c=1}^{C} V_c Y_{x,c} \neq 0\right] = 0.$$

And thus we also have that:

$$\mathop{\mathbb{P}}_{(x,y)\sim\mathcal{D}_S}\left[H(x) = \vartheta\right] = \mathop{\mathbb{P}}_{(x,y)\sim\mathcal{D}_S}\left[\sum_{c=1}^{C} V_c Y_{x,c} = 0\right].$$

□

Note that, in the very unlikely event that, in one of the iterations of TripletBoost, the selected triplet classifier has an error of exactly 1/2, Theorem 5 might not hold for our method.




(a) With respect to $k$ and $\beta$.

(b) With $C = \frac{n(n-1)}{2}$.

Figure 4: Illustration of the bound in Theorem 4 when $n = 100$.

# G Illustration of Theorem 4

When $n = 100$ we illustrate the bound obtained in Theorem 4 in Figure 4. This figure shows that the transition between abstaining and non-abstaining classifier depends on both the proportion of triplets available and the number of classifiers considered. In particular, when the number of combined classifiers increases, one needs a smaller number of triplets. Conversely when the number of triplets available increases, one can consider combining fewer classifiers.

# H Experiments

In Figures 5, 6, 7, and 8 we provide the plots omitted in the main paper.

# I Details on the Movielens experiment

As a proof of concept we considered the 1m movielens dataset (Harper and Konstan, 2016). This dataset contains 1 million ratings from 6040 users on 3706 movies and each movie has one or several genres (there is 18 genres in total). To demonstrate the interest of our approach we proposed (i) to use the users' ratings to obtain some triplets of the form movie $m_i$ is closer to movie $m_j$ than to movie $m_k$, and (ii) to use TripletBoost to learn a classifier predicting the genres of the movies.

To generate the triplets we propose to consider that movie $m_i$ is closer to $m_j$ than to $m_k$ if, on average, users that rated all three movies rated $m_i$ and $m_j$ more similarly than $m_i$ and $m_k$. The underlying intuition is that users like and dislike genres of movies — for example a user that dislikes horror movies but likes comedy movies will probably give low ratings to The Ring (2002) and Scream (1996) and a higher rating to The Big Lebowsky (1998). Formally let $r_{u,i}$ be the rating of user $u$ on movie $m_i$ and $r_{u,i,j} = |r_{u,i} - r_{u,j}|$ then the triplet set $T$ is

$$T = \left\{ (m_i, m_j, m_k) : \left( \sum_{u \in U_{i,j,k}} \frac{\mathbb{I}_{r_{u,i,j} < r_{u,i,k}} - \mathbb{I}_{r_{u,i,j}} > |r_{u,i,k}|}{|U_{i,j,k}|} \right) > 0 \right\}$$

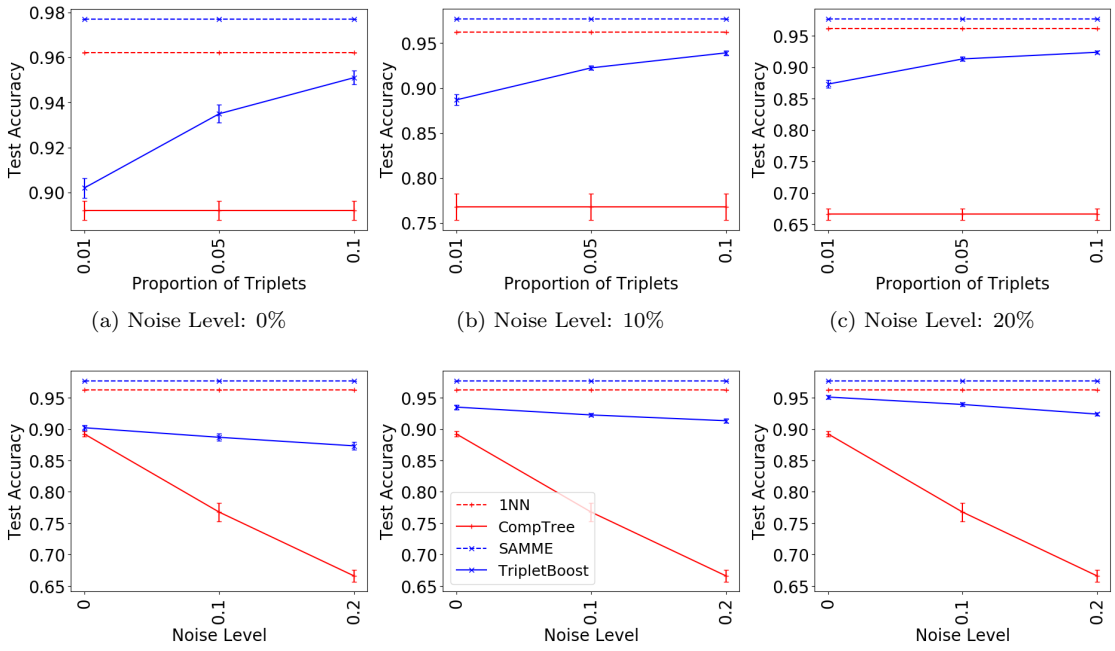
where $U_{i,j,k}$ is the set of users that rated all three movies. Each user has only rated a small number of movies and might give a high, respectively low, rating to a movie with a genre that he usually rates lower, respectively higher. Thus we only have access to a noisy subset of all the possible triplets.

We used a random sample of 2595 movies, and their corresponding triplets, to learn a multi-label classifier using TripletBoost (with $10^6$ boosting iterations). Since we are in a multi-label

setting we would like to predict how relevant each genre is for a new movie rather than a single genre. To obtain such a quantity we can simply ignore the arg max in Equation (5) in the main paper to obtain a classifier $H(m, y)$ that predicts the weight of a genre $y$ for a movie $m$:

$$H(m, y) = \sum_{c=1}^{C} \alpha_c \mathbb{I}_{h_c(m) \neq \vartheta \wedge y \in h_c(m)}.$$

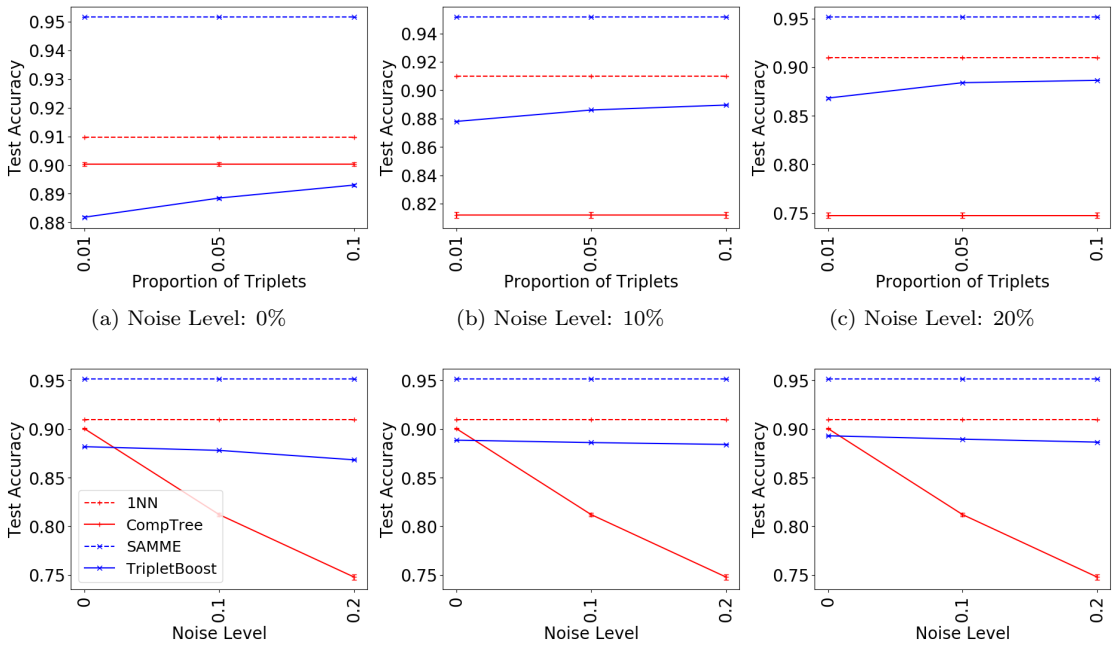
In the test phase we used the 1111 remaining movies to measure the performance of the learned classifier. First we considered the precision of the genre predicted with the highest weight and obtained a value of 0.83168. It means that in $\sim$83% of the cases the genre predicted with the highest weight is correct. We also considered the recall of the 5 genres predicted with the highest weights and obtained a value of 0.92943. It implies that, on average, the 5 genres predicted with the highest weights cover $\sim$92% of the genres of the considered movie.




(a) Noise Level: 0% (b) Noise Level: 10% (c) Noise Level: 20%

(d) Proportion of Triplets: 1% (e) Proportion of Triplets: 5% (f) Proportion of Triplets: 10%

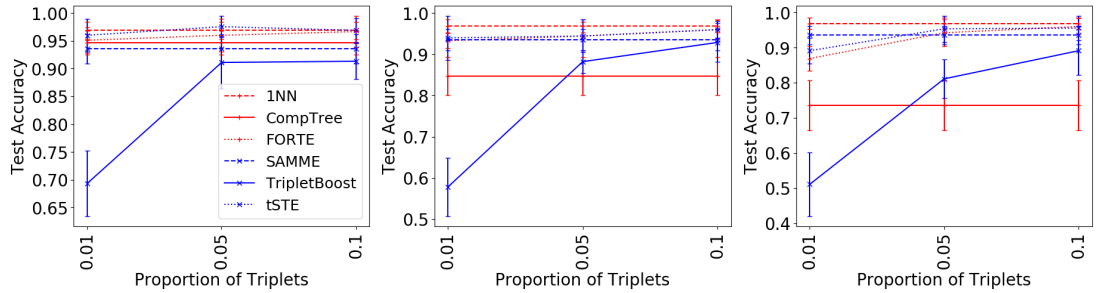
Figure 5: Results on the Gisette dataset. In the top row we vary the noise level from 0 to 20%. In the bottom row we vary the proportion of triplets available from 1 to 10%.



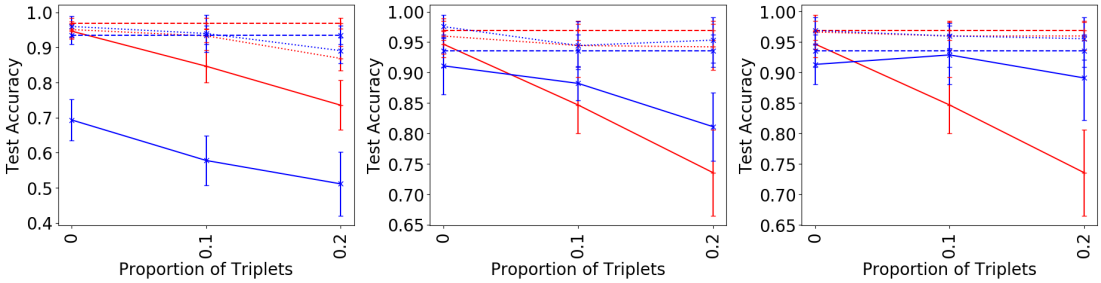

(a) Noise Level: 0% (b) Noise Level: 10% (c) Noise Level: 20%

(d) Proportion of Triplets: 1% (e) Proportion of Triplets: 5% (f) Proportion of Triplets: 10%

Figure 6: Results on the CodRna dataset. In the top row we vary the noise level from 0 to 20%. In the bottom row we vary the proportion of triplets available from 1 to 10%.

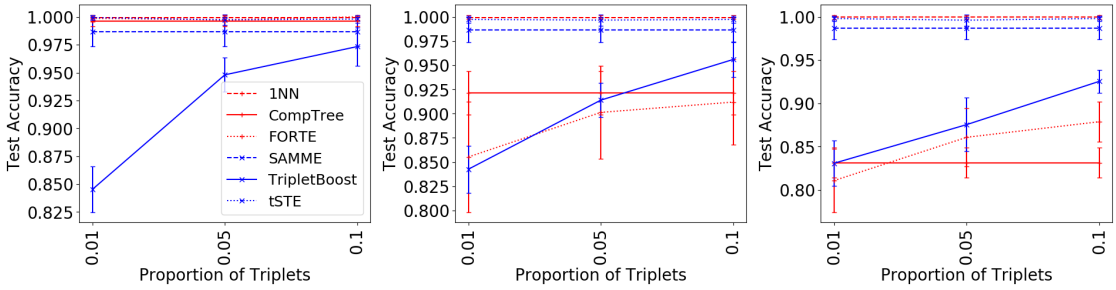



(a) Noise Level: 0% (b) Noise Level: 10% (c) Noise Level: 20%

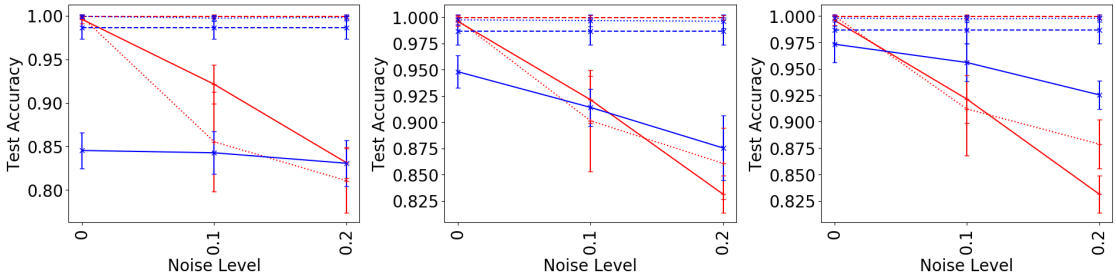



(d) Proportion of Triplets: 1% (e) Proportion of Triplets: 5% (f) Proportion of Triplets: 10%

Figure 7: Results on the Iris dataset. In the top row we vary the noise level from 0 to 20%. In the bottom row we vary the proportion of triplets available from 1 to 10%.




(a) Noise Level: 0% (b) Noise Level: 10% (c) Noise Level: 20%




(d) Proportion of Triplets: 1% (e) Proportion of Triplets: 5% (f) Proportion of Triplets: 10%

Figure 8: Results on the Moons dataset. In the top row we vary the noise level from 0 to 20%. In the bottom row we vary the proportion of triplets available from 1 to 10%.