A new F -score gradient-based training rule for the linear model

Mariusz Paradowski¹ · Michał Spytkowski¹ · Halina Kwaśnicka¹

Received: 2 September 2014 / Accepted: 8 August 2017 / Published online: 30 August 2017
© The Author(s) 2017. This article is an open access publication

Abstract Delta rule is a standard, well-established approach to train perceptron recognition model. However, mean squared error, on which it is based, is not suitable estimate for some problems, like information retrieval or automatic data annotation. F -score, a combination of precision and recall, is one of the major quality measures and can be used as an alternative. In this paper we present perceptron training model based on f -score. An approximate of f -score is proposed, based on components which are both continuous and differentiable. It allows to formulate a gradient-descent training routine, conceptually similar to the standard delta rule.

Keywords Perceptron · Activation function · Gradient descent · Automatic annotation

1 Introduction

Increasing the amount of electronic data creates the need for their automated analysis. Machine learning is the research area that deals with the development and testing of new methods for automatic data analysis. Object recognition is a subproblem of machine learning. It is useful in solving many interesting and difficult real-life problems [1]. One of such real-life problems is automatic data annotation—a multi-class and multi-label object recognition problem. To make it

even more difficult, automatic data annotation problems are often considered as high-class imbalance problems.

In this paper we address the basic recognition model—the **linear perceptron**. On top of it, many other, more complex solutions may be proposed. The presented research is done from the perspective of automatic data annotation.

1.1 Linear recognition models

Training of linear models has a long history. One shall note the classic Fisher's *Linear Discriminant Analysis* (LDA, e.g., [2]). Existence of closed-form, analytical solution is the largest advantage of discriminant analysis (both linear and quadratic). A disadvantage of linear discriminant analysis is the assumption on equality of covariance matrices for both classes. Also, it can cause typical difficulties related to zeroed or near-zeroed generalized variance [6] and covariance matrix inverse problems, especially for data with a large number of attributes. One of the possible solutions is to filter out attributes with zeroes-related eigenvalues [6]. Another possible solution is the use *Regularized Discriminant Analysis* (RDA) [3]. The basic assumption is that some recognition problems may be *ill-posed* due to insufficient amount of data comparing to the number of attributes. It combines together covariance matrix, diagonal variance matrix and identity matrix and thus makes the training process solvable. An interesting solution for LDA covariance matrix calculation is given by Fukunaga [4]. It combines both covariance matrices using *weighted average* instead of *average*, as originally proposed by Fisher. An extension of LDA is Kernel-LDA [5], which uses *kernel trick* known from *Support Vector Machines* to address linearly non-separable problems.

The second family of approaches to train linear models is *logistic regression* (e.g., [6]). Logistic regression is

✉ Mariusz Paradowski
mariusz.paradowski@pwr.edu.pl

¹ Department of Computational Intelligence, Wrocław University of Science and Technology, Wyb. Stanisława Wyspiańskiego 27, 50-370 Wrocław, Poland

formulated using *odds ratio*. Unlike LDA, it does not have a closed-form solution. *Expectation Maximization* method is used to estimate parameters of the model. A detailed discussion and comparison of LDA and logistic regression is given (among others) by Press and Wilson [7].

Another family of linear model training approaches is related to *Rosenblatt perceptron*. Frean [8] points out two families of methods for perceptron training. The first one is based on the *pocket algorithm* [9]. It uses an additional set of weights to stabilize the training process in case of linearly non-separable data. Pocket algorithm has been extended to multi-class problems by introducing *entropy* as the quality criterion [10]. The extended algorithm is used to induce decision trees, where multi-class splits are frequent. The second methods family is based on *gradient-based optimization*. Typically, a perceptron model is trained using *generalized delta rule* (e.g., [11]) and its extensions, e.g., multiplicative update rule [12] with faster convergence. Alternative training approaches focus on different quality measures. Crammer and Singer [13] present a method for linear model training based on various quality measures known from information retrieval, e.g., *average precision* or *f-score*. The presented training routine is generic in nature, but the update function is not gradient based. Mencia and Furnkranz [14, 15] extend the above idea and discuss simultaneous training of all perceptrons used in the decision model. Collins [16] proposed a new variant of perceptron algorithm developed for tagging problems. The algorithm is justified through a modification of the proof of convergence of the perceptron algorithm for classification problems.

1.2 Automatic data annotation and *f-score* measure

Let us now discuss the research from the perspective of automatic data annotation. As already mentioned earlier, it is a multi-class, multi-label problem with possible class imbalances. It is widely accepted that *accuracy* is not the best choice when addressing the problem.

Precision, *recall* and *f-score* quality measures are often chosen instead, e.g., [17–23]. Precision represents how well a classifier works, when it recognizes a class. Recall shows how well a classifier works, when it should recognize a class. Downside of precision and recall is that they have to come together, which makes quality comparison difficult. *F-score* quality measure solves the problem. *F-score* can be easily used in tasks of machine learning where the calculation of its gradient is not required. Many examples of non-gradient *f-score* optimization applications may be found, e.g., Gunes et al. [24] proposed a multi-class *f-score* feature selection. *F-score* is used as a measure of discriminating power of features in the classification of two-class pattern recognition problems.

As the proposed approach has its roots in automatic data annotation, we should consider *precision*, *recall* and *f-score*

as a quality measure [17–23]. Other typical features of automatic data annotation are large dictionaries and high-class imbalance. Quality of each class is evaluated separately; overall results are averaged throughout all classes. Another important property is that for each class annotation evaluation is measured mostly using positive responses of the given class. Negative responses are taken into consideration through the denominator of precision.

1.3 Contribution

The key goal and contribution of this paper is to propose a **new perceptron training rule based on *f-score* quality measure**. *F-score* quality measure is chosen because it is widely accepted in automatic data annotation research (see Sect. 1.2). *F-score* is a harmonic mean of precision and recall, which aggregates classifier answers during testing. The measure is not continuous, nor differentiable, thus it is not appropriate for gradient-based training. To solve the problem we propose an **approximation of *f-score*** which is both continuous and differentiable. Additionally, we provide a short theoretical analysis of the recall quality measure (the component of *f-score* as mentioned above) in the context of perceptron training. We show that an *approximation of recall* leads to a training rule identical to the *weighted delta rule*. In consequence, the proposed approximation of *f-score* can be considered as belonging to the same family of approaches as the *weighted delta rule*.

The paper is organized as follows. Next section contains the problem definition together with symbol definition. The third section presents the proposed approach. The fourth section compares *weighted delta rule* and the proposed *f-score* rule. The fifth section demonstrates a practical example. The last section summarizes the paper.

2 The standard recognition model

In this section we formalize the standard *perceptron* recognition model. We define symbols, functions and quality measures necessary for further discussion.

2.1 Recognition model

Let us define a binary recognition problem. The input data are defined as $\mathbf{x} = (x_1, x_2, \dots, x_d)$, where d is the dimensionality of the feature vector. The output classes are $O = \{a, b\}$. The *perceptron* recognition model Ψ is built on top of a standard linear approximation function and is defined as follows:

$$\Psi(\mathbf{x}) = f\left(\gamma \sum_{i=0}^d w_i x_i\right), \quad (1)$$

where $\gamma > 0$ is the scaling factor, $\mathbf{w} = [w_0, w_1, \dots, w_d]$ is a predefined weight vector, w_0 is the bias value, and $x_0 = -1$ is the bias-related fixed value. We require that the *activation function* f would meet the following properties:

$$f \in C^1, \quad \forall_{x < y}, f(x) < f(y), \quad (2)$$

$$\lim_{x \rightarrow \infty} f(x) = h_f < \infty, \quad \lim_{x \rightarrow -\infty} f(x) = l_f > -\infty. \quad (3)$$

The real-valued recognition model $\Psi(\mathbf{x})$ may be converted into a binary decision model $\Psi_D(\mathbf{x})$ by introducing a decision threshold f_t dependent on the chosen activation function f :

$$\Psi_D(\mathbf{x}) : R^d \rightarrow O, \quad \Psi_D(\mathbf{x}) = \begin{cases} a & \text{if } \Psi(\mathbf{x}) \geq f_t \\ b & \text{if } \Psi(\mathbf{x}) < f_t \end{cases}. \quad (4)$$

2.2 Training process

The goal of the training process is to estimate the vector of weights $\mathbf{w}$ (including the bias) given a set of labeled training data X . We assume that the labeling y_x of a training example $\mathbf{x} \in X$ is one of two possible classes: $a \in O$ or $b \in O$. The training set X consists of two disjoint subsets. Subset A contains instances labeled by class a , subset B contains instances labeled by class b :

$$X = A \cup B, \quad A \cap B = \emptyset. \quad (5)$$

Labeling y_x associated with the training examples $\mathbf{x} \in X$ should be correlated with the used activation function f . To simplify calculations, but without losing generality, we assume the following properties of the activation function f :

$$y_x = h_f = 1 : \mathbf{x} \in A, \quad y_x = l_f = -1 : \mathbf{x} \in B, \quad f_t = 0. \quad (6)$$

An exemplary activation function, which follows the above is $f \equiv \tanh$. Other functions (e.g., *sigmoid*) may be rescaled accordingly. Perceptron Ψ error may be estimated using *mean squared error* over labeled training set X :

$$\text{MSE}(X) = \frac{1}{|X|} \sum_{\mathbf{x} \in X} (y_x - \Psi(\mathbf{x}))^2. \quad (7)$$

Thus, the training process requires solving of an optimization problem defined as:

$$\Psi^* = \arg \min_{\Psi} \sum_{\mathbf{x} \in X} (y_x - \Psi(\mathbf{x}))^2. \quad (8)$$

One of the major advantages of the training process is the existence of error function derivatives. It enables the usage of a gradient-descent method for efficient training, i.e., the well-known *delta rule*.

2.3 Training with unbalanced datasets

The major problem with the above model is that it does not handle unbalanced datasets well. In real-world data training examples are usually highly unbalanced. Class instance ratios are lower than $\frac{|A|}{|B|} = 0.01$ in: medical data, image-related data, multi-class datasets where a stands for a single class and b for all other classes. At least three different solutions may be considered to remedy this situation:

1. Training set subsampling,
2. Training set supersampling,
3. Training examples weighting.

Let us consider the third option, because it naturally fits into *mean squared error* training. Eq. 8 comes in the weighted form, which makes both classes equally important:

$$\Psi_A^* = \arg \min_{\Psi} \sum_{\mathbf{x} \in X} \alpha(\mathbf{x}) (y_x - \Psi(\mathbf{x}))^2, \quad (9)$$

where

$$\alpha(\mathbf{x} \in A) = \frac{1}{|A|}, \quad \alpha(\mathbf{x} \in B) = \frac{1}{|B|}. \quad (10)$$

Above symbols and recognition model become our reference point for further discussion.

3 Perceptron with alternative quality measures

Intense research in information retrieval brought to attention alternative recognition quality measures with interesting properties. *Recall* r_k , *precision* p_k and their combination *f-score* f_k are worth mentioning. They are all per-class quality measures, where $k \in \{a, b\}$ is the chosen class:

$$r_k = \frac{c_k}{e_k}, \quad p_k = \frac{c_k}{g_k}, \quad f_k = \frac{2p_k r_k}{p_k + r_k} = \frac{2c_k}{e_k + g_k}, \quad (11)$$

where c_k is the number of correctly recognized objects of class k , g_k is the number of objects recognized as class k , e_k is the size of set K ($K = A$ or $K = B$) representing examples of class k :

$$c_k = \sum_{\mathbf{x} \in K} |\{\Psi_D(\mathbf{x})\} \cap \{k\}|, \quad g_k = \sum_{\mathbf{x} \in X} |\{\Psi_D(\mathbf{x})\} \cap \{k\}|, \quad e_k = |K|. \quad (12)$$

Above elementary measures (Eq. 12) are defined as integer numbers, which makes them unsuitable for gradient training approach. Let us now propose three approaches to perceptron training. The first one is *recall*-based training. We show that it is equivalent to *weighted delta rule*. The second one is

f-score-based training, focused on both classes. The last one and the most important is *f-score*-based training but focused only on a single class.

3.1 Recall-based training

The first presented approach is based on recall. Recall of class $k \in \{a, b\}$ is maximal when the number of correctly recognized objects of class k is equal to the total number of objects of class k . But it is also maximal when all test examples are classified as class k . This makes it unsuitable as a quality criterion for training. However, if we use recall of both classes a and b , the above problem disappears [25]. The recognition quality criterion may be defined as:

$$r_a + r_b = \frac{c_a}{e_a} + \frac{c_b}{e_b} = \frac{e_a - i_a}{e_a} + \frac{e_b - i_b}{e_b}, \quad (13)$$

where i_k is the number of incorrectly recognized objects of class k :

$$i_k = \sum_{\mathbf{x} \in K} (1 - |\{\Psi_D(\mathbf{x})\} \cap \{k\}|). \quad (14)$$

Recall is a discrete quality measure; thus, the above criterion is not continuous, nor differentiable. To adapt it to gradient-based methods, we propose an *approximated recall* $\hat{r}_k$ quality measure. To formulate approximated recall $\hat{r}_k$ we need to define its approximated components. Size of the set e_k is a constant, thus $\hat{e}_k = e_k$. The number of incorrectly recognized objects i_k may be approximated by the *mean squared error*:

$$\hat{i}_k = \frac{1}{4} \sum_{\mathbf{x} \in K} (y_{\mathbf{x}} - \Psi(\mathbf{x}))^2. \quad (15)$$

Usage of above approximation has two interesting properties. First, the approximated recall $\hat{r}_k$ is equal to recall r_k , if the scaling factor $\gamma \rightarrow \infty$ (see Eq. 1, smooth activation function f tends to signum function):

$$\lim_{\gamma \rightarrow \infty} \hat{r}_k = r_k. \quad (16)$$

Second, approximated recall training criterion becomes equivalent with the weighted delta rule (Eqs. 9 and 15):

$$\begin{aligned} \hat{r}_a + \hat{r}_b &= \frac{e_a - \frac{1}{4} \sum_{\mathbf{x} \in A} (y_{\mathbf{x}} - \Psi(\mathbf{x}))^2}{e_a} + \frac{e_b - \frac{1}{4} \sum_{\mathbf{x} \in B} (y_{\mathbf{x}} - \Psi(\mathbf{x}))^2}{e_b} \\ &= \frac{e_a}{e_a} + \frac{e_b}{e_b} - \frac{1}{4} \sum_{\mathbf{x} \in A} \frac{1}{e_a} (y_{\mathbf{x}} - \Psi(\mathbf{x}))^2 - \frac{1}{4} \sum_{\mathbf{x} \in B} \frac{1}{e_b} (y_{\mathbf{x}} - \Psi(\mathbf{x}))^2 \\ &= 2 - \frac{1}{4} \sum_{\mathbf{x} \in X} \alpha(\mathbf{x}) (y_{\mathbf{x}} - \Psi(\mathbf{x}))^2, \end{aligned} \quad (17)$$

where

$$\alpha(\mathbf{x} \in A) = \frac{1}{e_a}, \quad \alpha(\mathbf{x} \in B) = \frac{1}{e_b}. \quad (18)$$

Thus, both approaches to training (see Eq. 9) are equal:

$$\Psi_r^* = \arg \max_{\Psi} (\hat{r}_a + \hat{r}_b) = \arg \min_{\Psi} \sum_{\mathbf{x} \in X} \alpha(\mathbf{x}) (y_{\mathbf{x}} - \Psi(\mathbf{x}))^2 = \Psi_{\Delta}^*. \quad (19)$$

The above result is important because of two reasons. It shows that the proposed recall approximation $\hat{r}_k$ is valid and shows the close relation between recall and weighted accuracy.

3.2 F-score-based training

The second presented approach is based on *f-score*. It combines both precision and recall. Unlike recall of class $k \in \{a, b\}$, *f-score* of class k is a sufficient training quality criterion. This gives us two possibilities: to focus on *f-score* of both classes a and b or to focus on *f-score* of a single, chosen class. The first approach seems to be more intuitive, but the second one has more practical sense. Let us now present the first approach, the second one is discussed in next section. The overall recognition quality of classes a and b may be formulated as (see Eqs. 12 and 14 for definition of components):

$$f_a + f_b = \frac{2c_a}{e_a + g_a} + \frac{2c_b}{e_b + g_b} = \frac{2e_a - 2i_a}{e_a + g_a} + \frac{2e_b - 2i_b}{e_b + g_b}. \quad (20)$$

F-score f_k is also a discrete quality measure and has to be approximated by $\hat{f}_k$. To define approximated *f-score* $\hat{f}_k$ we also need to define approximated number of recognized objects $\hat{g}_k$ (see Eq. 12). We formulate $\hat{g}_k$ as an aggregated answer of perceptron Ψ . Thus, $\hat{g}_a$ and $\hat{g}_b$ are defined as:

$$\hat{g}_a = \frac{1}{4} \sum_{\mathbf{x} \in X} (1 + \Psi(\mathbf{x}))^2, \quad \hat{g}_b = \frac{1}{4} \sum_{\mathbf{x} \in X} (1 - \Psi(\mathbf{x}))^2. \quad (21)$$

Finally, approximated *f-score* of class $\hat{f}_k$ is given as:

$$\hat{f}_k = \frac{8e_k - \sum_{\mathbf{x} \in K} (y_{\mathbf{x}} - \Psi(\mathbf{x}))^2}{4e_k + \sum_{\mathbf{x} \in X} (1 \pm \Psi(\mathbf{x}))^2}. \quad (22)$$

The approximation $\hat{f}_k$ is equal to f_k when $\gamma \rightarrow \infty$ (see Eq. 1):

$$\lim_{\gamma \rightarrow \infty} \hat{f}_k = f_k. \quad (23)$$

Now, training quality criterion given by Eq. 20 may be approximated as (see Eqs. 9, 15 and 18):

$$\hat{f}_a + \hat{f}_b = \frac{8e_a - \sum_{\mathbf{x} \in A} (y_{\mathbf{x}} - \Psi(\mathbf{x}))^2}{4e_a + \sum_{\mathbf{x} \in X} (1 + \Psi(\mathbf{x}))^2} + \frac{8e_b - \sum_{\mathbf{x} \in B} (y_{\mathbf{x}} - \Psi(\mathbf{x}))^2}{4e_b + \sum_{\mathbf{x} \in X} (1 - \Psi(\mathbf{x}))^2}. \quad (24)$$

The training process can now be defined as an optimization problem:

$$\Psi_f^* = \arg \max_{\Psi} (\hat{f}_a + \hat{f}_b). \quad (25)$$

Given that the optimized parameters of recognition function Ψ are weights $\mathbf{w}$, the above function is differentiable and thus applicable for gradient-descent optimization:

$$\begin{aligned} \frac{\partial(\hat{f}_a + \hat{f}_b)}{\partial w_i} = & \frac{2 \sum_{\mathbf{x} \in A} \frac{\partial \Psi(\mathbf{x})}{\partial w_i} (y_{\mathbf{x}} - \Psi(\mathbf{x}))}{4e_a + \sum_{\mathbf{x} \in X} (1 + \Psi(\mathbf{x}))^2} + \frac{2 \sum_{\mathbf{x} \in B} \frac{\partial \Psi(\mathbf{x})}{\partial w_i} (y_{\mathbf{x}} - \Psi(\mathbf{x}))}{4e_b + \sum_{\mathbf{x} \in X} (1 - \Psi(\mathbf{x}))^2} + \\ & - \frac{\left(2 \sum_{\mathbf{x} \in A} (1 + \Psi(\mathbf{x})) \frac{\partial \Psi(\mathbf{x})}{\partial w_i}\right) [8e_a - \sum_{\mathbf{x} \in A} (y_{\mathbf{x}} - \Psi(\mathbf{x}))^2]}{(4e_a + \sum_{\mathbf{x} \in X} (1 + \Psi(\mathbf{x}))^2)^2} + \\ & - \frac{\left(2 \sum_{\mathbf{x} \in B} (1 - \Psi(\mathbf{x})) \frac{\partial \Psi(\mathbf{x})}{\partial w_i}\right) [8e_b - \sum_{\mathbf{x} \in B} (y_{\mathbf{x}} - \Psi(\mathbf{x}))^2]}{(4e_b + \sum_{\mathbf{x} \in X} (1 - \Psi(\mathbf{x}))^2)^2}, \end{aligned} \quad (26)$$

where (assuming Eq. 6 and without losing generality):

$$\frac{\partial \Psi(\mathbf{x})}{\partial w_i} = \Phi(\mathbf{x}) \gamma x_i = \left[1 - \tanh \left(\gamma \sum_{i=0}^d w_i x_i \right) \right]^2 \gamma x_i. \quad (27)$$

The proposed weight update rule is applicable only for batch training, incremental training is not possible.

3.3 Single class f -score-based training

The last and the key approach is the single class f -score training. We show that we can train a binary classifier focusing only on a single class (either a or b). The method can be applied for training various real-life problems, such as automatic data annotation. Usage of the method is suggested when successful recognition of chosen class is much more important than recognition of the other class. Additionally, it releases us from the need of class weights configuration (Eqs. 9 and 10). It also fine tunes for a broadly accepted quality criterion. A practical example is shown in Sect. 5.

To simplify notation, we choose class a , but presented approach is identical for class b . The quality criterion is defined as:

$$\Psi_F^* = \arg \max_{\Psi} \hat{f}_a. \quad (28)$$

Training examples and classifier answers for class a are directly taken into consideration. An important note should be made here regarding instances of class b . Despite, f -score is defined only for class a , training instances for class b are also taken into consideration in the form of *false positive* responses which are counted in the precision component of f -score. Thus, indirect training using b class examples also takes place. Gradient update rule in a class f -score $\hat{f}_a$ training is defined as (see Eqs. 24 and 26):

$$\begin{aligned} \frac{\partial \hat{f}_a}{\partial w_i} = & \frac{2 \sum_{\mathbf{x} \in A} \frac{\partial \Psi(\mathbf{x})}{\partial w_i} (y_{\mathbf{x}} - \Psi(\mathbf{x}))}{4e_a + \sum_{\mathbf{x} \in X} (1 + \Psi(\mathbf{x}))^2} + \\ & - \frac{\left(2 \sum_{\mathbf{x} \in A} (1 + \Psi(\mathbf{x})) \frac{\partial \Psi(\mathbf{x})}{\partial w_i}\right) [8e_a - \sum_{\mathbf{x} \in A} (y_{\mathbf{x}} - \Psi(\mathbf{x}))^2]}{(4e_a + \sum_{\mathbf{x} \in X} (1 + \Psi(\mathbf{x}))^2)^2}. \end{aligned} \quad (29)$$

All above quality criteria may be used in the same training procedure, shown below.

3.4 Training process

The training process consists of two parts. The first part is a random search for an acceptable solution. The second part is a gradient optimization routine. The method is generic in nature; all above quality criteria can be used. Thus, quality measure q can be defined as:

- $q = \Delta$ (Eq. 9) which represents the weighted MSE,
- $q = f$ (Eq. 25) which represents the model proposed in Sect. 3.2,
- $q = F$ (Eq. 28), which represents the model proposed in Sect. 3.3.

The training pseudocode is presented as Algorithm 1. Please note that the chosen quality measure q is used in both parts of the method.

Algorithm 1 Recognition method training process

Require: $q(\cdot)$ – quality measure
Require: X – training set
Require: $rand$ – number of random search iterations,
Require: $grad$ – number of gradient descent iterations,
Require: β_0 – initial value of gradient step size,
Require: w_{min}, w_{max} – weight range for randomized search,
Require: ϵ – minimum normalized difference between solutions,
Ensure: $\mathbf{w}^*$ – vector of optimized weights

```

1:  $\mathbf{w}^* \leftarrow [1]^{d+1}$  {Initial solution}
2: for  $i = 1 \rightarrow rand$  do
3:    $\mathbf{w} \leftarrow \mathcal{U}(w_{min}, w_{max})^{d+1}$  {Randomize using uniform distribution}
4:   if  $q(\mathbf{w}) > q(\mathbf{w}^*)$  then
5:      $\mathbf{w}^* \leftarrow \mathbf{w}$  {Better solution found, update solution}
6:   end if
7: end for
8:  $\beta \leftarrow \beta_0$  {Set initial step size}
9:  $q_0 \leftarrow q(\mathbf{w}^*), q_1 \leftarrow q_0$  {Quality of initial and "first better" solution}
10: repeat
11:    $\mathbf{w} \leftarrow \mathbf{w}^* + \beta \frac{\partial q(\mathbf{w}^*)}{\partial \mathbf{w}^*}$  {Optimize using gradient descent}
12:    $q_p \leftarrow q(\mathbf{w}^*)$  {Memorize previous solution quality}
13:   if  $q(\mathbf{w}) > q(\mathbf{w}^*)$  then
14:      $q_1 \leftarrow q(\mathbf{w}^*) : q_1 = q_0$  {Memorize first better solution}
15:      $\beta \leftarrow \beta_0, \mathbf{w}^* \leftarrow \mathbf{w}$  {Update solution, reset step}
16:   else
17:      $\beta \leftarrow \frac{1}{2} \beta$  {No improvement, reduce step}
18:   end if
19:    $grad \leftarrow grad - 1$ 
20: until  $grad < 0$  or  $0 < q(\mathbf{w}^*) - q_p < \epsilon(q_1 - q_0)$ 
21: return  $\mathbf{w}^*$ 

```

A classic *gradient-descent* routine is used as the optimization routine. Step size β is dynamically adjusted to satisfy the optimization improvement criterion $q(\mathbf{w}) > q(\mathbf{w}^*)$,

where $\mathbf{w}$ represents current solution and $\mathbf{w}^*$ —current best solution. There are two stop criteria of the gradient optimization method:

1. Maximum number of gradient-descent iterations is equal to *grad*.
2. Improvement between the current and the previous solutions $q(\mathbf{w}^*) - q_p$ has to be larger than a given threshold ϵ . To become independent of the quality function and the dataset, the threshold is normalized and is dependent of the first gradient improvement $q_1 - q_0$.

It is worth noting that the above training procedure is one of many possible and is given mostly for reference purposes. Gradient-descent routines with other stop criteria or step update strategies can also be used. It is also possible to skip the random search and start gradient descent from predefined weights. Such approach is used in the practical example shown in Sect. 5.

4 Experimental verification

Experimental verification of the proposed recognition model has two separate goals and in consequence consists of two parts. Both goals focus on recognition quality; however, the basic definition of experiments is different:

1. In the first experiment we directly evaluate the proposed linear model. Our goal is to present the distribution of quality differences and test if these differences are significant or not. For this purpose we use a series of randomly generated datasets based on various data distributions.
2. In the second experiment we address a real-life automatic data annotation problem. The proposed model is used within a complex multi-class, multi-label recognition model. The experiment is presented in Sect. 5.

4.1 Quality evaluation of the proposed method

In this section we describe the quality evaluation of the proposed perceptron training approach. Our goal is to verify, if the proposed approach achieves better results than the baseline approach. Two types of training are compared:

1. Proposed single class f -score training (see Sect. 3.3),
2. Classic weighted delta rule (see Sect. 2.3).

The baseline method for all presented tests is the weighted delta rule. The motivation of the baseline selection relates to the discussion in Sects. 1.3 and 3.1. The proposed training

approach may be seen as the extension of the weighted delta rule. It is shown that a simpler variant of the proposed method (approximation of recall) gives the training rule identical to the weighted delta rule. In consequence, both methods can be considered as belonging to the same family of approaches.

Quality of the solution is measured using f -score. Necessary equations of f -score are defined by Eqs. 11 and 12.

Three major tests are done, one for each dataset type. Five predefined class a and b ratios are tested. All datasets are randomly generated, according to assumed probability distributions. In each test 20 datasets are generated. For each dataset training and testing under different initialization conditions is repeated 50 times. Thus, each configuration is tested 1000 times.

Algorithm 1 is used for training in both approaches. Identical initialization conditions are used for each method. The only difference is the optimized function. First we randomly seek for the acceptable solution. Later we employ gradient-descent optimization from the best one found. Fixed number of iterations is used in both cases.

4.2 Dataset generation

The first part of the quality evaluation is done on randomly generated datasets. Let us first define necessary symbols:

- $\mathcal{N}(\mu, \Sigma)$ represents a *multivariate normal distribution*, where μ is the mean vector and Σ is the covariance matrix.
- $\mathcal{U}(u_1, u_2)$ represents a *single dimensional continuous uniform distribution*, where all data points are distributed between u_1 and u_2 .
- $\mathcal{D}(u_1, u_2)$ represents a *single dimensional discrete uniform distribution*, where all data points are distributed between u_1 and u_2 .

Three types of datasets are taken into consideration for the training process verification. All datasets have predefined distributions of classes a and b training samples. Dimensionality of all datasets is random and has an uniform distribution:

$$d \sim \mathcal{D}(2, 20). \quad (30)$$

All datasets share a predefined ratio of training examples $\frac{|A|}{|B|}$.

The ratio is given as a parameter and is equal to: 100, 10, 1, 0.1, 0.01.

4.2.1 Randomized linearly separable datasets

The first type of datasets are randomized datasets with linearly separable means (LS). Elements of these datasets are

drawn from a *normal distribution*. Recognition of these datasets is easiest, i.e., a simple hyperplane should correctly separate at least half of both classes. Datasets are defined as follows:

$$X = A \cup B, \quad A \sim \mathcal{N}(\mu^A, \Sigma^A), \quad B \sim \mathcal{N}(\mu^B, \Sigma^B), \quad (31)$$

where parameters of the distributions are also random ($i, j \in \{1, \dots, d\}$):

$$\mu_i^A \sim \mathcal{U}(-1, 1), \quad \mu_i^B \sim \mathcal{U}(-1, 1), \quad (32)$$

$$\sigma_{ii}^A \sim \mathcal{U}(0, 1), \quad \sigma_{ii}^B \sim \mathcal{U}(0, 1), \quad (33)$$

$$\sigma_{ij}^A = 0, \quad \sigma_{ij}^B = 0, \quad i \neq j. \quad (34)$$

4.2.2 Randomized linearly non-separable datasets

The second type of datasets are randomized datasets with linearly non-separable means (LNS). Elements of these datasets are drawn from a *mixture of multivariate normal distributions*. These datasets are much more difficult for recognition using linear model. Given k_A and k_B mixture components, the data distribution is given as:

$$X = A \cup B, \quad A \sim \sum_{l=1}^{k_A} \frac{1}{k_A} \mathcal{N}(\mu_l^A, \Sigma_l^A), \quad B \sim \sum_{l=1}^{k_B} \frac{1}{k_B} \mathcal{N}(\mu_l^B, \Sigma_l^B), \quad (35)$$

where parameters of all distributions are random. The number of mixture components is given as:

$$k_A \sim \mathcal{D}(2, 5), \quad k_B \sim \mathcal{D}(2, 5). \quad (36)$$

Mixture component distributions ($l \in \{1, \dots, k\}$ and $i, j \in \{1, \dots, d\}$) are given as:

$$\mu_{l,i}^A \sim \mathcal{U}(-1, 1), \quad \mu_{l,i}^B \sim \mathcal{U}(-1, 1), \quad (37)$$

$$\sigma_{l,ii}^A \sim \mathcal{U}(0, 1), \quad \sigma_{l,ii}^B \sim \mathcal{U}(0, 1), \quad (38)$$

$$\sigma_{l,ij}^A = 0, \quad \sigma_{l,ij}^B = 0, \quad i \neq j. \quad (39)$$

4.2.3 Randomized uniform datasets

The third type of datasets are randomized datasets (RND). Elements of these datasets are drawn from an *uniform distribution*. These datasets are especially interesting, because they represent noise without any dominant pattern. In consequence, recognition methods should be unable to generalize the data in any meaningful way and will work mostly on predefined a priori clauses. Uniform datasets are defined as follows:

$$X = A \cup B, \quad A \sim \mathcal{U}(-1, 1), \quad B \sim \mathcal{U}(-1, 1). \quad (40)$$

All three types of datasets are used in the experiments.

4.3 Method parameter setup

There are two important parameters of the proposed method. The first one is activation function scaling parameter γ (see Eq. 1). The second one is optimization method step β (see Algorithm 1). To get the quality comparison as fair as possible we decide to choose method parameters randomly. In each test different values of parameters are used. Parameter values are drawn from a *uniform distribution*. Distributions defined as follows:

$$Pr(\gamma) = \mathcal{U}(1, 50), \quad Pr(\alpha) = \mathcal{U}(0, 1), \quad \beta = 10^{3\alpha-3} \in (0.001, 1). \quad (41)$$

Initial tests of the proposed f -score delta rule and classic delta rule show that the second method prefers smaller values of β . Thus, to make the comparison as fair as possible, distribution of β parameter is uniform in logarithmic scale. As a result, both methods are given as far as possible equal number of parameter values preferable ranges. Usage of non-rescaled uniform distribution of β parameter has also been tested. Quantitatively they are slightly different, but qualitatively they remain the same.

4.4 Comparison of recognition quality

This section presents the comparison of recognition quality on artificially generated datasets. Both methods are evaluated multiple times. This allows to form a *cumulative distribution function* and to perform *statistical hypothesis testing*. Each randomly generated dataset is split in half. First 50% of the dataset are used as the training subset. Remaining 50% of the dataset are used as the test subset. Due to the nature of the dataset generation process, both subsets share the same distribution.

Due to a very large number of test results (3 dataset types, 5 class ratios, 20 datasets for each class ratio, 50 repetitions), statistical data are shown. For each type of dataset and class ratio a *cumulative distribution function* (CDF) is presented. The probability density function is defined as $Pr(Q(\Psi_F^*) - Q(\Psi_A^*))$, where the quality measure Q is *single class f-score*, see Eq. 11 (the exact one, not the approximate used during the training process). Thus, all results greater than 0 are in favor of the proposed method. All results less or equal than 0 are in favor of the original weighted delta rule.

The first test is performed on the linearly separable dataset. This is a very important test, because the recognition model is also linear. Thus, the proposed recognition model is designed to handle such data. Quality comparison between

Fig. 1 Cumulative distribution function of $Pr(Q(\Psi_F^*) - Q(\Psi_\Delta^*))$ for 20×5 randomly generated linearly separable datasets, 50 repetitions on each dataset

Fig. 2 Cumulative distribution function of $Pr(Q(\Psi_F^*) - Q(\Psi_\Delta^*))$ for 20×5 randomly generated nonlinearly separable datasets, 50 repetitions on each dataset

Fig. 3 Cumulative distribution function of $Pr(Q(\Psi_F^*) - Q(\Psi_\Delta^*))$ for 20×5 randomly generated datasets, 50 repetitions on each dataset

the proposed training and weighted delta rule is shown in Fig. 1. The proposed approach outperforms the weighted delta rule when the dataset imbalance is high.

The second test is performed on linearly non-separable datasets. It is a much more difficult case; both recognition

models are unable to correctly separate data. Quality comparison between the proposed training and weighted delta rule is shown in Fig. 2. The improvement in case of these datasets is more frequent and larger in value.

The last test is performed on data generated using uniform distribution, thus without any predefined pattern. The goal of this test is to verify how well the linear model fits the quality criterion in case there are no observable clues. Quality comparison between the proposed training and weighted delta rule is shown in Fig. 3. Large improvement may be observed in case class a examples are dominant. When class b examples dominate the dataset, quality is similar.

4.5 Concluding remarks

Experimental verification on three types of generated datasets is shown. Two training approaches are compared: delta rule and proposed approximated f -score rule. Recognition quality is measured using exact f -score quality measure.

Statistical hypothesis testing is performed to compare both methods. *Correlated samples t test* (e.g., [11]) is used to check if one method outperforms the other. The t value in t test is formulated as:

$$t = \frac{E[Q(\Psi_F^*) - Q(\Psi_\Delta^*)]}{\sqrt{\text{Var}[Q(\Psi_F^*) - Q(\Psi_\Delta^*)]}} \sqrt{n-1}, \quad (42)$$

where $Q(\Psi_F^*)$ is a random variable representing quality measurement of the proposed single class f -score training, $Q(\Psi_\Delta^*)$ is a random variable representing quality measurement of the reference classic weighted delta rule, n is the number of quality measurements ($n = 20 \times 50 = 1000$). Test significance level is set to 0.05. Thus for a sample of $n = 1000$ measurements, confidence intervals are the following. F -score training is significantly better than the reference model if $t \in \langle 1.64; \infty \rangle$. Weighted delta rule is significantly better than f -score training if $t \in (-\infty; -1.64)$. In all other cases, there is the lack of resolve. Test results are presented in Table 1.

Statistical hypothesis tests show a significant predominance of the proposed training approach over the classic weighted delta rule. The largest quality improvement is observed for per-class example ratio $\frac{|A|}{|B|} > 1$. At the same time standard deviation is decreased; thus, the training process is more stable. Given enough data, approximate f -score rule fine tunes split hyperplane for class a , with small regard for class b . Classic delta rule always tries to find a compromise between classes a and b . Due to prominence of class b , delta rule generates split hyperplane more fine tuned for the class.

Table 1 Comparison of both methods using statistical hypothesis testing

Dataset	$ A / B $	$Q(\Psi_F^*)(\%)$	$Q(\Psi_d^*)(\%)$	Difference (%)	t test	Winner
LS	0.01	29.6±16.7	28.5±21.9	1.1±17.5	1.97	F-Delta
LS	0.1	67.7±15.6	66.8±17.4	0.9±7.0	4.21	F-Delta
LS	1	88.7±6.4	90.3±6.6	-1.6±1.9	-26.72	Weighted MSE
LS	10	96.5±1.3	94.3±4.7	2.2±4.0	17.05	F-Delta
LS	100	99.1±0.5	95.3±4.5	3.9±4.4	27.47	F-Delta
NLS	0.01	6.4±3.3	5.0±2.3	1.3±2.7	15.66	F-Delta
NLS	0.1	31.8±7.5	31.5±7.1	0.2±3.0	2.48	F-Delta
NLS	1	71.0±3.3	68.8±6.6	2.2±4.4	16.11	F-Delta
NLS	10	93.5±2.0	80.3±8.3	13.2±7.8	53.43	F-Delta
NLS	100	97.5±2.1	81.1±9.0	16.4±8.8	58.55	F-Delta
RND	0.01	1.4±1.1	1.9±0.4	-0.6±1.2	-14.74	Weighted MSE
RND	0.1	16.0±0.6	15.0±1.3	0.9±1.4	20.91	F-Delta
RND	1	66.6±0.1	49.2±7.1	17.5±7.1	77.46	F-Delta
RND	10	95.2±0.0	64.3±13.5	31.0±13.5	72.43	F-Delta
RND	100	99.5±0.0	70.1±14.8	29.4±14.8	62.64	F-Delta

Various generated dataset types and class example ratios are used. Recognition results of a class; f -score quality measurement is used. Proposed method significantly outperforms weighted MSE in 13 out of 15 cases

Other scenarios are observed for $\frac{|A|}{|B|} < 1$. Estimation of split hyperplane is much more difficult for both methods due to the insufficient amount of class a data. Quality differences are smaller, but test results are above the significance threshold. As a result the proposed f -score approach is still dominant. The most difficult case is the nonlinearly separable datasets for $\frac{|A|}{|B|} \ll 1$. Generation of good split hyperplane is practically not possible, due to dataset properties.

An interesting result is achieved for *uniformly random* (RND) datasets with example ratio $\frac{|A|}{|B|} = 1$. Weighted delta rule achieves f -score value equal to 49.2%. It is an intuitive result, because the method weights both classes equally. Proposed F-Delta method is clearly biased toward recognition of class a and achieves f -score equal to 66.6%.

To sum up, if both classes are of equal importance, classic delta rule should be chosen. However, if one class is of larger importance than the other, approximate f -score rule may be taken into consideration.

5 Practical example—complex recognition problem

The proposed approach has its roots in *automatic image annotation*, where precision, recall and f -score are used as a quality measure [17–23]. Large dictionaries and high-class imbalance are typical to automatic annotation. Quality of each class is evaluated separately; overall results are

averaged throughout all classes. Another important property is that for each class annotation evaluation is measured mostly using *positive* responses of the given class. *Negative* responses are also taken into consideration only through the *denominator of precision*.

Let us now define necessary symbols. Formally, we have redefine annotation problem as a set of binary subproblems. We focus on annotation of a single, chosen class. Positive instances of the class are denoted as class a . Negative instances of the class are denoted as b . The labeled training set Z for the chosen class may be defined as:

$$Z = \{(\mathbf{z}_1; c_1), (\mathbf{z}_2; c_2), \dots, (\mathbf{z}_n; c_n)\}, \quad (43)$$

where $\mathbf{z}_i$ is a data vector representing i -th training example and $c_i \in \{a, b\}$ is the associated binary decision.

5.1 PATSI—baseline annotation model

The inspiration of the presented research is *Photo Annotation Through Similar Images (PATSI)* automatic image annotator [19–21]. We present it shortly for clarity, using the above formalism. PATSI is a nearest-neighbor-based method. The input of the method is a feature vector $\mathbf{q}$ (*query*). The output is a binary decision a or b . We assume the existence of some function $d(\cdot, \cdot)$ capable of measuring distances between feature vectors $\mathbf{z}_i$ and $\mathbf{q}$. We use *Bag of Words* feature vectors distance [26].

The first tier of PATSI transforms input feature vector $\mathbf{q}$ into distance-based feature vector $\mathbf{q}_d$. Let us define an ordering $\mathbf{y}$ of elements in the training set Z based on the distance function $d(\cdot, \cdot)$:

$$\mathbf{y} = [y_1, y_2, \dots, y_n] : d(\mathbf{q}, \mathbf{z}_{y_i}) \leq d(\mathbf{q}, \mathbf{z}_{y_{i+1}}), i \in \{1, \dots, n-1\}. \quad (44)$$

The output of first tier $\mathbf{q}_d$ is defined using ordering $\mathbf{y}$. It is a vector of m (m is a parameter) binary features:

$$\mathbf{q}_d = [q_{y_1}, q_{y_2}, \dots, q_{y_m}] : m \ll n, \quad q_{y_i} = \begin{cases} 1 & \text{if } c_{y_i} = a \\ 0 & \text{if } c_{y_i} = b \end{cases}. \quad (45)$$

The second tier of PATSI aggregates results of the first tier. It is a simple linear model with *signum* activation function. Weights of linear model are fixed and equal to $w_i = \frac{1}{i}$. The only configurable weight is the bias t . The recognition model Ψ_p is defined as:

$$P(\mathbf{q}_d) = \sum_{i=1}^m w_i q_{y_i} = \sum_{i=1}^m \frac{q_{y_i}}{i}, \quad \Psi_p(\mathbf{q}_d) = \begin{cases} a & \text{if } P(\mathbf{q}_d) \geq t \\ b & \text{if } P(\mathbf{q}_d) < t \end{cases}. \quad (46)$$

5.2 F-Delta—PATSI with gradient-based training

Our intention is to make the PATSI training process formally more elegant, effective and general. We extended PATSI by introducing a new gradient-based training. First tier of PATSI is without any change. The second tier uses the proposed linear model, with training based on single class f -score.

Instead of decision model from (Eq. 46), we use the perceptron (Eq. 1). This makes the training process much more flexible. Linear model weights do not have to be fixed any more, they can be trained. To address the annotation quality measurement, we choose single class f -score as training quality criterion. Output of the linear model provides the final decision of the method.

The training process is performed according to Algorithm 1. Original PATSI weights (Eq. 46) are the initial solution. We skip the random search and only perform gradient descent. Maximum number of gradient-descent iterations is set to 1000. ϵ is set to 0.1; thus, minimum acceptable improvement is equal to 0.1 of the first improvement in gradient descent. Activation function scaling parameter γ is set to 20.

5.3 Recognition results

Verification of recognition quality of the proposed approach is done on four annotated image datasets: *ICPR-2004*¹, *MGV-2006*, *Matching*² and *Lower Silesia 10K*³. Datasets differ in a number of ways: image domain, number of images,

number of classes. Statistics of image datasets are given together with the recognition results in Table 2. Test protocol is identical for all datasets. 10-fold cross-validation is used to estimate recognition quality. The quality measure is f -score, typical to automatic image annotation problems. Quality is measured for 10, 20, 50 and 100 of the most frequent and for all words in the training set. We have tested and compared the following approaches:

1. Original PATSI with manually set threshold,
2. PATSI with MSE gradient-based training (MSE-PATSI),
3. PATSI with LDA training with simple feature selection (LDA-PATSI),
4. PATSI with the proposed f -score training (F-Delta-PATSI).

Gradient-descent step parameter β has been tuned separately for both gradient training approaches. Initial tests show that MSE-based training achieves best results for $\beta = 0.0005$, f -score approach achieves best results for $\beta = 0.025$.

Table 2 presents a comparison of annotation quality achieved by the proposed approach (F-Delta) and by reference approaches. The proposed training rule outperforms the original PATSI approach in all cases. Due to gradient-based training, F-Delta is able to fine tune to the data. Unlike PATSI, gradient descent allows simultaneous training of all weights and bias. It also outperforms MSE-based training in almost all cases, because it directly optimizes the measured quality criterion. LDA-based solution works well (2 best results out of 4 tests) for most frequent words, because it has enough data to properly estimate covariance matrices of both classes. For less frequent words proper estimation of positive class covariance is much more difficult. It is worse than original PATSI approach in almost all cases.

6 Summary

In this paper a new perceptron training rule is proposed. The rule is based on single class f -score and is applicable if following criteria are met:

- Precision, recall or f -score are appropriate for quality evaluation,
- High-class imbalance is present in the data,
- Recognition quality of chosen class is much more important than the opposite one.

The method works well if the number of instances of a chosen class is much less than the number of instances of the second class. For such datasets achieved f -score results are much higher, compared to the ones achieved by classic

¹ <http://www.cs.washington.edu/research/imagedatabase/groundtruth>.

² <http://www.visible.iipwr.wroc.pl>, download section.

³ Contains images from <http://zabytkidolnegoslaska.com.pl>.

Table 2 Recognition results of the proposed approach (F-Delta), weighted mean squared error (MSE), linear discriminant analysis (LDA) and original PATSI on various annotated image databases

Most freq. Words	10	20	50	100	All
ICPR-2004 dataset, 1109 images, 322 words					
Word freq.	0.254	0.165	0.086	0.049	0.017
F-Delta-PATSI	0.594	0.516	0.507	0.424	0.212
MSE-PATSI	0.588	0.511	0.505	0.422	0.211
LDA-PATSI	<i>0.576</i>	<i>0.492</i>	<i>0.433</i>	<i>0.337</i>	<i>0.135</i>
Original PATSI	0.586	0.502	0.502	0.419	0.210
MGV-2006 dataset, 721 images, 73 words					
Word freq.	0.291	0.197	0.098	μ	0.068
F-Delta-PATSI	0.564	0.515	0.378	μ	0.298
MSE-PATSI	0.563	0.511	0.377	μ	0.292
LDA-PATSI	0.566	<i>0.494</i>	<i>0.324</i>	μ	<i>0.253</i>
Original PATSI	0.549	0.495	0.366	μ	0.290
Matching dataset, 100 images, 29 words					
Word freq.	0.097	0.069	μ	μ	0.054
F-Delta-PATSI	0.465	0.463	μ	μ	0.331
MSE-PATSI	0.462	0.454	μ	μ	<i>0.313</i>
LDA-PATSI	0.453	<i>0.385</i>	μ	μ	<i>0.273</i>
Original PATSI	0.453	0.447	μ	μ	0.320
Lower Silesia dataset, 10797 images, 273 words					
Word freq.	0.219	0.153	0.080	0.045	0.018
F-Delta-PATSI	0.559	0.528	0.461	0.439	0.366
MSE-PATSI	0.564	0.521	<i>0.450</i>	<i>0.431</i>	0.365
LDA-PATSI	0.576	0.525	<i>0.377</i>	<i>0.267</i>	<i>0.191</i>
Original PATSI	0.544	0.516	0.455	0.435	0.364

Best results are shown in bold; results worse than original PATSI are shown in italic

weighted delta rule. Thus, the proposed method is an attractive tool for training highly unbalanced datasets with focus on f -score instead of accuracy, especially for automatic image annotation problems.

Acknowledgements The work is partially financed from the European Union structural funds, Innovative Economy Operational Program, POIG.01.01.02-14-013/09. Lower Silesia image dataset contains images downloaded from <http://zabytkidolnegoslaska.com.pl>. We thank the author of the web page for sharing the collection of photographs.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

- Murphy Kevin P (2012) Machine learning a probabilistic perspective. MIT Press, Cambridge
- Therrien CW (1989) Decision estimation and classification: an introduction to pattern recognition and related topics. Wiley, Hoboken
- Friedman JH (1989) Regularized discriminant analysis. J Am Stat Assoc 84(405):165–175
- Fukunaga K (1990) Introduction to statistical pattern recognition. Academic press, Cambridge
- Mika S, Ratsch G, Weston J, Scholkopf B, Mullers KR (1999) Fisher discriminant analysis with Kernels. In: Neural networks for signal processing IX, 1999. Proceedings of the 1999 IEEE signal processing society Workshop, pp 41–48
- Johnson RA, Wichern DW (2007) Applied multivariate statistical analysis, 6th edn. Prentice Hall, Upper Saddle River
- Press S (1978) James, Wilson, S.: choosing between logistic regression and discriminant analysis. J Am Stat Assoc 73(364):699–705
- Frean M (1992) A “thermal” perceptron learning rule. Neural Comput 4(6):946–957
- Gallant SI (1990) Perceptron-based learning algorithms. IEEE Trans Neural Netw 1(2):179–191
- Sethi Ishwar K, Jae H (1994) Design of multicategory multi-feature split decision trees using perceptron learning. Pattern Recognit 27(7):939–947

11. Kuncheva Ludmila I (2004) Combining pattern classifiers: methods and algorithms. Wiley, Hoboken
12. Kivinen J, Warmuth MK (1997) Exponentiated gradient versus gradient descent for linear predictors. *Inf Comput* 132(1):1–63
13. Crammer K, Singer Y (2003) A family of additive online algorithms for category ranking. *J Mach Learn Res* 3:1025–1058
14. Loza Mencia E, Furnkranz J (2008) Pairwise Learning of Multilabel Classifications with Perceptrons. *Neural Networks*
15. Fürnkranz J et al (2008) Multilabel classification via calibrated label ranking. *Mach Learn* 73(2):133–153
16. Collins M (2002) Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In: *EMNLP 02: proceedings of the ACL-02 conference on empirical methods in natural language processing*, vol 10, pp 1–8
17. Cooper M, Foote J, Girgensohn A, Wilcox L (2005) Temporal event clustering for digital photo collections. *ACM Trans Multimed Comput Commun Appl (TOMCCAP)* 1(3):269–288
18. Demner-Fushman D, Antani S, Simpson M, Thoma GR (2009) Annotation and retrieval of clinically relevant images. *Int J Med Inf* 78:59–67
19. Maier O, Stanek M, Kwasnicka H (2010) PATSI—Photo annotation through similar images with annotation length optimization. *Intelligent Inf Syst*. pp 219–232
20. Stanek M, Broda B, Kwasnicka H (2010) Patsi—photo annotation through finding similar images with multivariate gaussian models. *Computer vision and graphics*. Springer, Berlin Heidelberg
21. Maier O, Kwasnicka H, Stanek M (2012) Image auto-annotation with automatic selection of the annotation length. *J Intell Inf Syst* 39(3):651–685
22. Xirong Li, et al. (2006) Image annotation by large-scale content-based image retrieval. In: *Proceedings of the 14th annual ACM international conference on multimedia*. ACM
23. Ameesh M, Pavlovic V, Kumar S (2008) A New Baseline for Image Annotation. *Computer Vision-ECCV 2008*. Springer, Berlin Heidelberg.
24. Gunes S, Polat K, Yosunkaya S (2010) Multi-class f-score feature selection approach to classification of obstructive sleep apnea syndrome. *Expert Syst Appl* 37:998–1004
25. Kwasnicka H, Paradowski M (2008) Resulted word counts optimization—a new approach for better automatic image annotation. *Pattern Recognit* 41(12):3562–3571
26. Arandjelovic R, Zisserman A (2012) Three things everyone should know to improve object retrieval, computer vision and pattern recognition (CVPR). In: *IEEE conference on*, pp 2911–2918