Multiclass Classification: Margins Revisited, Novelty Detection via Trees, and Visual Object Recognition with Sparse Features Derived from a Convolution Neural Net

by

Lior Bar

under the supervision of

Prof. Daphna Weinshall

a thesis submitted in partial fulfillment of the
requirements for the degree of
Master of Science

at the School of Computer Science and Engineering
Hebrew University of Jerusalem, Israel 91904
June, 2015

Abstract

This thesis comprises three nearly self contained parts. First we examine a few types of multi-class Support Vector Machine (SVM) classifiers that are typically used in applied machine learning. Unlike the original binary SVM formulation, in these classifiers the margins which are being maximized in the optimization problem do not represent distances to the decision boundaries of the final classifier. We investigate whether improvement can be obtained by employing classifiers which maximiz margins with respect to the classifier's actual decision boundaries. Perhaps surprisingly, we will prove a theorem that negates that theory - the optimization problem solved by the unified versions (Crammer & Singer, 2001), (Weston & Watkins, 1998), obtains a solution that is identical to that of the optimization problem that maximizes margins with respect to the actual decision boundaries. In addition, we present a connection between this version and the 1-vs-1 SVM multiclass classifier.

Later, we explore the use of descriptors extracted from pre-trained CNNs for image classification of new classes; in our work we addressed the sparsity of those descriptors. With CNN features, we observed that for 1-vs-Rest, the use of binary descriptors (by quantizing the CNN features) yields comparable results to the use of the full feature value. Whereas, for Nearest Neighbor and Image Retrieval, the binary descriptors improve classification results.

Finally, we examine hierarchical tree-like meta-structures describing a set of classes and discover that the learnt classification trees resemble those reported by human observers. We sought to use these trees for information transfer to new classes of object, where the task is to recognize the novelty of a sample and use the tree to bootstrap the classification of new classes.

למידת תיוג רב מחלקתית: בחינה מחדש של מקסום שולי ההפרדה, זיהוי מחלקות חדשות בעזרת עצים וזיהויי אובייקטים ע"פ תמונות בעזרת ייצוג דליל שהופק מרשת נוירונים מלאכותית

ליאור בר

הוגש כמילוי חלקי של החובות לתואר מוסמך במדעים

מנחה: פרופ' דפנה וינשל

תמוז תשע"ה

בית הספר להנדסה ומדעי המחשב
על שם רחל וסלים בנין
האוניברסיטה העברית בירושלים
ישראל

תקציר

התזה מורכבת משלושה חלקים כמעט בלתי תלויים. בחלק הראשון אנו מסתכלים על השימוש במכונת וקטורים תומכים לצורך למידת סיווג מחלקות. הבחנו שבניגוד למקרה הבינארי, המסווג בשיטת אחד-נגד-השאר וגרסתיו המאוחדות, לא מנסים, בבעיית המקסימיזציה שלהם, למקסם את השוליים למפריד שנוצר בפועל בתהליך הלמידה. לאחר שיצרנו מסווג שממקסם את השוליים האפקטיביים, להפתעתנו, הוכחנו משפט שמראה שהמסווג שנוצר, שקול למסווג שנוצר בגרסאות המאוחדות של אחד-נגד-השאר. בהמשך הצגנו גם קשר בין המסווג הנ"ל למסווג ה- אחד-נגד-אחד.

בחלק השני, בחנו את השימוש במתארי תמונות שהוצאו מרשת נוירונים שאומנה מראש כדי לתייג קבוצה חדשה של מחלקות. התייחסנו לדלילות הייצוג של המתארים הנ"ל וראינו שבשימוש בשיטת אחד-נגד-השאר ביצוע בינאריזציה של המתארים (כל מה שלא אפס הוא אחד) נותן תוצאות ברות השוואה לשימוש בכל טווח הערכים של המתארים. בנוסף הראנו שעבור שיטת הסיווג "השכן הקרוב" ועבור הבאת תמונות דומות, הבינאריזציה אפילו משפרת את תוצאת הסיווג.

בחלק האחרון ייצרנו מבנה עץ היררכי על המחלקות, הבחנו שבשיטות שייצרנו, העצים שמתקבלים דומים לעצים שבני אדם יוצרים. מטרתנו הייתה לאפשר זיהוי של מחלקות חדשות שלא נראו בשלב האימון ושינוי אזורים בעץ ע"פ המחלקה החדשה שזוהתה לצורך זיהוייה בעתיד.

Acknowledgments

I would like to thank my advisor, Prof. Daphna Weinshall. Prof. Weinshall took an active part in the research. I have been privileged to have her support and inspiration.

In addition, I would like to thank Reuven Siman-Tov (prof. Weinshall’s student) who implemented the parallel running framework for the trees.

Last but certainly not least, I would like to thank my family and my girlfriend (Edna Zigdon) for their endless love and encouragement.

Tables of contents

Multiclass Support Vector Machines – Maximizing Margins of Decision Boundaries

Abstract

A few types of multi-class Support Vector Machine (SVM) classifiers are typically used in applied machine learning, including the 1-vs-Rest classifier and its unification to many classes. Unlike the original binary SVM formulation, in these classifiers the margins which are being maximized in the optimization problem do not represent distances to the decision boundaries of the final classifier. We investigate whether improvement can be obtained by employing classifiers which maximize margins with respect to the classifier's actual decision boundaries. Maybe surprisingly, we prove a theorem which states that the problems are equivalent – when solving the optimization problems which underlie the 2 most common unified versions of the 1-vs-Rest SVM classifier (Crammer & Singer, 2001),(Weston & Watkins, 1998), one obtains the same solutions as if optimization is sought using margins with respect to the classifier's actual decision boundaries. We also show that this classifier is equal to the 1-vs-1 multiclass classifier, when the latter is regularized in such a way that each binary separator is required to be the difference between two uni-class separators. These results may help to explain empirical observations where the different multi-class SVM classifiers perform rather similarly, with inconsistent differences.

1. Introduction

Multi-class classification is a learning problem in which the learner is trained to separate examples from k different labels. For the binary problem with k=2, one of the more effective methods is the Support Vector Machine (SVM) classifier. Using the hinge loss, the algorithm finds a decision boundary which separates the two classes while achieving the largest distance (margin) from the training examples.

A common way to create a multiclass SVM classifier from binary SVM classifiers is termed 1-vs-Rest SVM. The algorithm trains a uni-class separator for each class, which gives a collection of binary SVM classifiers that separate one class from the rest of the classes. In test time, a new point is assigned the label of the separator with the highest margin among all uni-class separators.

With more than 2 classes, this procedure appears to have at least one flaw: the margins which are maximized in the optimization problem when solving for the uni-class separators during training are not the margins with respect to the decision boundaries of the final classifier; this follows from the final voting stage over all uni-class separators. This problem is illustrated in Fig. 1 where both the uni-class separators of the 1-vs-Rest classifier and the final decision boundaries between the classes are shown to be different.

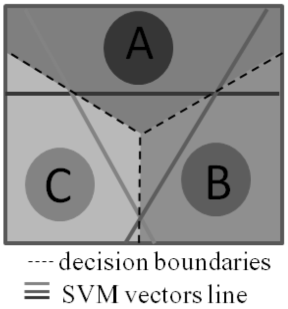


Figure 1. 1-vs-Rest: Decision boundaries vs. uni-class SVM vector lines. By definition, decision boundaries are the lines which segment the plane into distinct regions, such that all the points in each region are assigned to a single class by the final classifier.

There has been a lot of work on multi-class classifiers which we cannot review here, including the generalization of binary SVM to this problem. The two most common unification schemes of the 1-vs-Rest classifier are defined in Def. 2 in Section 2.2, which we denote by C&S (Crammer & Singer, 2001) and W&W (Weston & Watkins, 1998). Another commonly used classifier is the 1-vs-1 SVM, which is defined in Section 2.3. The 1-vs-1 classifier has a higher VC dimension than the other classifiers discussed here. Therefore, one may expect it to perform better, when given access to large training data.

(Hsu & Lin, 2002) empirically compared 1-vs-Rest SVM, 1-vs-1 SVM, C&S and W&W, using the linear kernel and a relatively low dimensional data (up to 19 attributes); in these experiments 1-vs-1 SVM outperformed the other methods. In

(Liu, et al., 2011), describing experiments with up to 255 attributes, 1-vs-1 also outperformed 1-vs-Rest and C&S in terms of accuracy, while C&S outperformed 1-vs-Rest. In contrast, the experiments described in (Gao & Koller, 2011) with the Caltech256 image dataset (Griffin, Holub, & Perona, 2006) using a high dimensional data representation (> 1000 attributes) and linear kernels, showed that 1-vs-Rest SVM can outperform 1-vs-1 SVM.

In the next chapter we investigate the “correct” optimization problem, which aims to maximize margins with respect to the actual decision boundaries.

Our main result proves that the solution to this problem is the same as the solution to the most common unified versions of the 1-vs-Rest SVM classifier (C&S and W&W). We also show the connection between this classifier and the 1-vs-1 SVM classifier for k classes; the two classifiers are equal when the $\frac{k(k-1)}{2}$ 1-vs-1 individual classifiers are required to be the difference between two classifiers from a set of k uni-class separators.

2. Extension of the binary SVM formulation to multi-class

Outline:

Section 2.1: starts by defining a multi-class SVM optimization problem which aims to maximize margins with respect to the actual decision boundaries of the final classifier.

Section 2.2 shows that the solution to this problem is the same as the solution to existing multi-class SVM formulations (Crammer & Singer, 2001)(Weston & Watkins, 1998).

Section 2.3 shows the equivalence of this classifier to the 1-vs-1 SVM classifier under some strong constraints.

Section 2.4 shows some empirical comparisons using the Caltech256 image dataset.

2.1 Multi-class SVM definition

Let $S = \{(\bar{x}_1, y_1), \ldots, (\bar{x}_m, y_m)\}$ denote a set of m training examples, where $\bar{x}_i \in \mathbb{R}^n$ and labels $y_i \in \{1, \ldots, k\}$. A multiclass classifier is a function $H: X \rightarrow Y$ that maps an instance $\bar{x}$ to label y. For simplicity of notation and without loss of generality, in the following discussion we ignore the bias terms in the definitions of the various SVM classifiers. It can be readily verified that all the results and proofs follow essentially as-is when bias is added to the classifiers.

Definition 1a: eSVM (SVM extended to multiclass, version 1):

$$H(\bar{x}) = \underset{r=[k]}{\operatorname{argmax}}\ \bar{w}_r\, \bar{x}$$

where $\bar{w}_{1,..,k} \in \mathbb{R}^n$ are obtained by solving:

$$\min_{\bar{w}_{1,..,k}, \varepsilon_{im} \geq 0}\ \frac{1}{2} \sum_{\substack{m,l=1 \\ m>l}}^{k} \|w_m - w_l\|_2^2 +\ C_1 \sum_{i=1}^{n} \sum_{m=1}^{k} \varepsilon_{im}$$

$$s.t. \forall i \epsilon [n]\ , \forall\ m \epsilon [k], m \neq y_i \colon (\ w_{y_i} x_i - w_m x_i) \geq 1 - \varepsilon_{im}$$

This definition uses the same soft constraints as (Weston & Watkins, 1998).

Definition 1b: eSVM (SVM extended to multiclass, version 2):

$$H(\bar{x}) = \underset{r=[k]}{\operatorname{argmax}} \overline{w}_r\, \bar{x}$$

where $\bar{w}_{1,..,k} \in \mathbb{R}^n$ are obtained by solving:

$$\min_{\bar{w}_{1,..,k}, \varepsilon_i \geq 0}\ \frac{1}{2} \sum_{\substack{m,l=1 \\ m>l}}^{k} \|w_m - w_l\|_2^2 +\ C_1 \sum_{i=1}^{n} \varepsilon_i$$

$$s.t. \forall i \epsilon [n]\ , \forall\ m \epsilon [k], m \neq y_i \colon (\ w_{y_i} x_i - w_m x_i) \geq 1 - \varepsilon_i$$

This definition uses the same soft constraints as (Crammer & Singer, 2001).

Note that both definitions are reduced to the usual SVM definition in the binary case $k = 2$, with weight vector $w =\ w_1 - w_2$. Note also that the constraints in Def. 2b can be written as follows:

$$\forall i \epsilon [n] :\ \underset{m \neq y_i}{min} (\ w_{y_i} x_i - w_m x_i) \geq 1 - \varepsilon_i$$

$$\forall i \epsilon [n] :\ \ w_{y_i} x_i - \underset{m \neq y_i}{max} (w_m x_i) \geq 1 - \varepsilon_i$$

Geometrically, $\left|\frac{w_{y_i}x_i - w_m x_i}{\left\|w_{y_i} - w_m\right\|_2}\right|$ is the distance between point x_i and the line defined by $(w_{y_i} - w_m)$, which is one of the actual decision lines of the classifier. Therefore, in the hard case of Def. 1b ($C_1 \to \infty$, $\varepsilon_i = 0$); using the constraints as written above, the classifier maximizes the margin between each training example x_i and the separator of the form $(w_{y_i} - w_m)$ which is closest to x_i. Hence, serves as the decision boundary for point x_i.

2.2 Equivalence to Commonly used Multi-class SVM Definitions

We recall the joint multi-class SVM definitions in use by the community:

Definition 2a: SVM extended to multiclass as in (Weston & Watkins, 1998) (W&W):

$$H(\bar{x}) = \underset{r=[k]}{\operatorname{argmax}}\, \overline{w_r}\, \bar{x}$$

where $\bar{w}_{1,..,k} \in \mathbb{R}^n$ are obtained by solving:

$$\min_{\bar{w}_{1,..,k},\varepsilon_{im}\geq 0} \; \frac{1}{2}\sum_{m=1}^{k}\|w_m\|_2^2 + C_2\sum_{i=1}^{n}\sum_{m=1}^{k}\varepsilon_{im}$$

$$s.t. \forall i\epsilon[n]\,, \forall\, m\epsilon[k], m \neq y_i\colon (\, w_{y_i}x_i - w_m x_i) \geq 1 - \varepsilon_{im}$$

Definition 2b: SVM extended to multiclass as in (Crammer & Singer, 2001) (C&S):

$$H(\bar{x}) = \underset{r=[k]}{\operatorname{argmax}}\, \bar{w}_r\, \bar{x}$$

where $\bar{w}_{1,..,k} \in \mathbb{R}^n$ are obtained by solving:

$$\min_{\bar{w}_{1,..,k},\varepsilon_{i}\geq 0} \; \frac{1}{2}\sum_{m=1}^{k}\|w_m\|_2^2 + \; C_2\sum_{i=1}^{n}\varepsilon_{i}$$

$$s.t. \forall i\epsilon[n], \forall\, m\epsilon[k], m \neq y_i\colon \;\; w_{y_i}x_i - w_m x_i \geq 1 - \varepsilon_{i}$$

Theorem 1: For $C = C_2 = \frac{C_1}{k}$ the classifiers obtained by Defs. 2a and 2b are equal to the eSVM classifiers using Defs. 1a and 1b respectively.

Proof: In the following we only prove the equivalence between Defs. 1b and 2b. Equivalence between Defs. 1a and 2a can be shown in the same manner.

Define $\forall\ i\epsilon\{2..k\}\ v_{1,i} = w_1 - w_i$. We start by changing variables from $\{\bar{w}_{1,..,k}\}$ to $\{\bar{w}_1, v_{1,2,..,k}\}$. Since the Jacobian of this variable transformation is 1, it will not change unconstrained extrema. For notation convenience we will also use additional notations for the following dependent variables $\forall i,j \in \{2..k\}$: $v_{i,j} = v_{1,j} - v_{1,i}$, so that $\forall i,j \in \{1..k\}$: $v_{i,j} = w_i - w_j$.

After this change of variables, we need to prove that:

$$\min_{\{\bar{w}_1, v_{1,2,..,k}\},\varepsilon_i\geq 0} \frac{1}{2}\{\|w_1\|_2^2 + \sum_{m=2}^{k}\left\|w_1 - v_{1,m}\right\|_2^2\} + \ C\sum_{i=1}^{n}\varepsilon_i$$

$$= \min_{v_{1,2,..,k},\varepsilon_i\geq 0} \frac{1}{2*k}\sum_{\substack{m,l=1\\ m>l}}^{k}\left\|v_{m,l}\right\|_2^2 + \ C\sum_{i=1}^{n}\varepsilon_i$$

$$s.t. \forall i\epsilon[n], \forall\ m\epsilon[k], m \neq y_i: \quad v_{y_i,m}x_i \geq 1 - \varepsilon_i$$

Lemma 1: for $v_{i,j} \in \mathbb{R}^n$ as defined above:

$$\sum_{j=2}^{k}\left\|v_{1,j}\right\|_2^2 - \frac{1}{k}(\sum_{j=2}^{k}v_{1,j})^2 = \frac{1}{k}\sum_{\substack{i,j=1\\ i<j}}^{k}\left\|v_{i,j}\right\|_2^2$$

Proof: by induction.

Basis: for $k = 3$

$$\frac{1}{3}\sum_{\substack{i,j=1\\ i<j}}^{3}\left\|v_{i,j}\right\|_2^2 = \frac{1}{3}\left\|v_{1,2}\right\|_2^2 + \frac{1}{3}\left\|v_{1,3}\right\|_2^2 + \frac{1}{3}\left\|v_{2,3}\right\|_2^2$$

$$= \frac{1}{3}\left\|v_{1,2}\right\|_2^2 + \frac{1}{3}\left\|v_{1,3}\right\|_2^2 + \frac{1}{3}\left\|v_{1,3} - v_{1,2}\right\|_2^2$$

$$= \left\|v_{1,2}\right\|_2^2 + \left\|v_{1,3}\right\|_2^2 - \frac{1}{3}\left\|v_{1,2}\right\|_2^2 - \frac{1}{3}\left\|v_{1,3}\right\|_2^2 - \frac{2}{3}v_{1,3} * v_{1,2}$$

$$= \sum_{j=2}^{3}\left\|v_{1,j}\right\|_2^2 - \frac{1}{3}(\sum_{j=2}^{3}v_{1,j})^2$$

Inductive step: assume for $k-1$, prove for k:

$$\sum_{j=2}^{k}\left\|v_{1,j}\right\|_2^2 - \frac{1}{k}\left(\sum_{j=2}^{k} v_{1,j}\right)^2 =$$

$$\left\|v_{1,k}\right\|_2^2 + \sum_{j=2}^{k-1}\left\|v_{1,j}\right\|_2^2 - \frac{1}{k}\left(v_{1,k} + \sum_{j=2}^{k-1} v_{1,j}\right)^2 =$$

$$\frac{k-1}{k}\left(\sum_{j=2}^{k-1}\left\|v_{1,j}\right\|_2^2 - \frac{1}{k-1}\left(\sum_{j=2}^{k-1} v_{1,j}\right)^2\right) + \left\|v_{1,k}\right\|_2^2 + \frac{1}{k}\sum_{j=2}^{k-1}\left\|v_{1,j}\right\|_2^2 - \frac{1}{k}\left\|v_{1,k}\right\|_2^2$$

$$-\frac{2}{k} v_{1,k}\sum_{j=2}^{k-1} v_{1,j}$$

Using the induction hypothesis:

$$\frac{1}{k}\left(\sum_{\substack{i,j=1\\ i<j}}^{k-1}\left\|v_{i,j}\right\|_2^2 + \left\|v_{1,k}\right\|_2^2 + (k-2)\left\|v_{1,k}\right\|_2^2 + \sum_{j=2}^{k-1}\left\|v_{1,j}\right\|_2^2 - 2v_{1,k}\sum_{j=2}^{k-1} v_{1,j}\right) =$$

$$\frac{1}{k}\left(\sum_{\substack{i,j=1\\ i<j}}^{k-1}\left\|v_{i,j}\right\|_2^2 + \left\|v_{1,k}\right\|_2^2 + \sum_{j=2}^{k-1}\left\|v_{1,k} - v_{1,j}\right\|_2^2\right) =$$

$$\frac{1}{k}\left(\sum_{\substack{i,j=1\\ i<j}}^{k-1}\left\|v_{i,j}\right\|_2^2 + \left\|v_{1,k}\right\|_2^2 + \sum_{j=2}^{k-1}\left\|v_{j,k}\right\|_2^2\right) =$$

$$\frac{1}{k}\sum_{\substack{i,j=1\\ i<j}}^{k}\left\|v_{i,j}\right\|_2^2$$

■

We shall now continue with proving that

$$\min_{\{\bar{w}_1,v_{1,2,..,k}\},\varepsilon_i\geq 0} \frac{1}{2}\{\|w_1\|_2^2+\sum_{m=2}^{k}\left\|w_1-v_{1,m}\right\|_2^2\}+\ C\sum_{i=1}^{n}\varepsilon_i$$

$$=\min_{v_{1,2,..,k},\varepsilon_i\geq 0} \frac{1}{2*k}\sum_{\substack{m,l=1\\m>l}}^{k}\left\|v_{m,l}\right\|_2^2+\ C\sum_{i=1}^{n}\varepsilon_i$$

$$s.t.\forall i\epsilon[n],\forall\, m\epsilon[k], m\neq y_i:\quad v_{y_i,m}x_i\geq 1-\varepsilon_i$$

We first observe that w_1 is unconstrained, and that only the left hand of the equation depends on it where

$$F=\|w_1\|_2^2+\sum_{i=2}^{k}\left\|w_1-v_{1,i}\right\|_2^2=\ k*\|w_1\|_2^2+\sum_{i=2}^{k}\left\|v_{1,i}\right\|_2^2-2w_1\sum_{i=2}^{k}v_{1,i}$$

Therefore, in any extremal point of this function, its derivative with respect to w_1 must be 0. Since the second derivative with respect to w_1 is positive, this extremum is a minimum. Hence, we can derive the value of w_1 at a minimum

$$0=\frac{\partial F}{\partial w_1}=2k*w_1-2\sum_{i=2}^{k}v_{1,i}\ \Rightarrow\quad \widehat{w}_1=\frac{1}{k}\sum_{i=2}^{k}v_{1,i}$$

Substituting $w_1=\widehat{w}_1$ into F we get:

$$F=\sum_{i=2}^{k}\left\|v_{1,i}\right\|_2^2+k\left(\frac{1}{k}\sum_{i=2}^{k}v_{1,i}\right)^2-\frac{2}{k}\left(\sum_{i=2}^{k}v_{1,i}\right)^2=\ \sum_{i=2}^{k}\left\|v_{1,i}\right\|_2^2-\frac{1}{k}\left(\sum_{i=2}^{k}v_{1,i}\right)^2$$

Using Lemma 1, it follows that at a minimum:

$$\sum_{m=1}^{k}\|w_m\|_2^2=F=\frac{1}{k}\sum_{\substack{i,j=1\\i<j}}^{k}\left\|v_{i,j}\right\|_2^2=\frac{1}{k}\sum_{\substack{m,l=1\\m>l}}^{k}\|w_m-w_l\|_2^2$$

It follows that the optimization problems defined in Def. 1b and Def. 2b are identical up to multiplication by k with the same constraints. Hence the classifiers are equal.

■

2.3 Relation to 1-vs-1 multi-Class SVM classifier

We recall another commonly used multi-class SVM classifier:

Definition 3: 1-vs-1 multiclass classifier

$$H(\bar{x}) = \underset{r=[k]}{\operatorname{argmax}} \sum_{l=1}^{k} \mathbf{1}_{\{\bar{w}_{r,l}\bar{x}>0\}}$$

$\bar{w}_{m,l} \in \mathbb{R}^n$ are binary classifiers between class m and l where $\bar{w}_{m,l} = -\bar{w}_{m,l}$, obtained by solving:

$$\min_{\bar{w}_{m,l},\varepsilon_{im}} \quad \frac{1}{2} \sum_{\substack{m,l=1\\ m>l}}^{k} \left\|\bar{w}_{m,l}\right\|_2^2 + C \sum_{i=1}^{n}\sum_{m=1}^{n} \varepsilon_{im}$$

$$s.t. \forall i\epsilon[n], \forall\, m\epsilon[k], m \neq y_i : \bar{w}_{y_i,m}\bar{x}_i \geq 1 - \varepsilon_{im}$$

Note that without additional constraints and when $C \to \infty$ (hard SVM), these are essentially $\frac{k(k-1)}{2}$ independent optimization problems defining each $\bar{w}_{m,l}$ independently.

Theorem 2: let $\bar{w}_{m,l} \in \mathbb{R}^n$ define a 1-vs-1 multiclass classifier, and let there be k vectors $\bar{w}_{1,..,k} \in \mathbb{R}^n$ such that $\bar{w}_{m,l} = \bar{w}_m - \bar{w}_l$. Under this constraint, the 1-vs-1 classifier is identical to the eSVM classifier defined by $\bar{w}_{1,..,k}$ (Def. 1a).

Proof: if we plug $\bar{w}_{m,l} = \bar{w}_m - \bar{w}_l$ into the 1-vs-1 definition above, the definition of $\bar{w}_{1,..,k} \in \mathbb{R}^n$ becomes identical to eSVM Def. 1a. Hence we need to show that

$$H(\bar{x}) = \underset{r=[k]}{\operatorname{argmax}} \sum_{l=1}^{k} \mathbf{1}_{\{\bar{w}_r\bar{x} > \bar{w}_l\bar{x}\}} = \underset{r=[k]}{\operatorname{argmax}}\ \bar{w}_r\,\bar{x}$$

using the fact the $\bar{w}_{r,l}\bar{x} > 0 \Rightarrow \bar{w}_r\bar{x} > \bar{w}_l\bar{x}$.

$(\Leftarrow) \quad \bar{r} = \underset{r=[k]}{\operatorname{argmax}}\ \bar{w}_r\,\bar{x} \quad \Rightarrow \quad \bar{w}_{\bar{r}}\,\bar{x} > \bar{w}_l\bar{x},\ \ \forall l \neq \bar{r}$

$$\Rightarrow \sum_{\substack{l=1\\ l\neq r}}^{k} \mathbf{1}_{\{\bar{w}_r\bar{x} > \bar{w}_l\bar{x}\}} \leq \sum_{\substack{l=1\\ l\neq \bar{r}}}^{k} \mathbf{1}_{\{\bar{w}_{\bar{r}}\bar{x} > \bar{w}_l\bar{x}\}} = k-1,\ \ \forall r \neq \bar{r}$$

$$\Rightarrow \bar{r} = \underset{r=[k]}{\operatorname{argmax}} \sum_{l=1}^{k} \mathbf{1}_{\{\bar{w}_r\bar{x} > \bar{w}_l\bar{x}\}}$$

(⇒) Let $\tilde{r} = \max_{r=[k]} \sum_{l=1}^{k} \mathbf{1}_{\{\bar{w}_r\bar{x} > \bar{w}_l\bar{x}\}}$. If $\tilde{r} = k - 1$ then this direction immediately follows. To prove this, we will assume that $\tilde{r} < k - 1$ and arrive at a contradiction. Let $\bar{r} = \underset{r=[k]}{\operatorname{argmax}} \sum_{l=1}^{k} \mathbf{1}_{\{\bar{w}_r\bar{x} > \bar{w}_l\bar{x}\}}$.

$$\tilde{r} < k - 1 \Rightarrow \exists\, r_0 \mid \bar{w}_{r_0}\bar{x} > \bar{w}_{\bar{r}}\bar{x}$$

$$\Rightarrow \bar{w}_{r_0}\bar{x} > \bar{w}_l\bar{x}, \qquad \{\forall l \neq \bar{r} \mid \bar{w}_{\bar{r}}\,\bar{x} > \bar{w}_l\bar{x}\}$$

$$\Rightarrow \sum_{l=1}^{k} \mathbf{1}_{\{\bar{w}_{r_0}\bar{x} > \bar{w}_l\bar{x}\}} \geq \sum_{l=1}^{k} \mathbf{1}_{\{\bar{w}_{\bar{r}}\bar{x} > \bar{w}_l\bar{x}\}} + 1$$

which contradicts the maximality of $\bar{r}$.

■

2.4 Multi-class classification with the Caltech256 database

To complete the discussion and in accordance with earlier empirical work described in the introduction, we tested 1-vs-Rest SVM, 1-vs-1 SVM, C&S (Crammer & Singer, 2001) and NN (Nearest Neighbor) on the Caltech256 dataset (Griffin, Holub, & Perona, 2006). We chose 5 random 60/20 train-test splits. For data representation we used the 20th layer of the pre-trained CNN Overfeat (Sermanet, et al., 2014) as image features (4096 features - ‘fast’ CNN version). For each train image we also added the horizontally mirrored image. All SVM and C&S classifiers were trained using LibLinear (Fan, Chang, Hsieh, Wang, & Lin, 2008) with a 3-fold cross validation to tune the C parameter ($C = 2^{-13}, 2^{-12} \ldots 2^{3}$). For Nearest Neighbor (NN) we use the MATLAB implementation. Results are shown below in Table 1.

Table 1: Classification methods

Caltech 256	1-vs-Rest	C&S	1-vs-1	NN
Test Error	34.63±1.0	33.91±0.6	33.33±0.9	53.73±0.3

It can be readily seen that in our experiments all SVM classifiers perform roughly the same with some edge to the 1-vs-1 classifier, while the NN classifier performs significantly worse.

We note in passing an empirical observation: when running a 1-vs-Rest SVM classifier, around 40% of the validation examples are considered negative by all of the uni-class separators; almost 60% are considered positive by only one SVM classifier.

3 Summary

In this chapter we took another look at the question of how to generalize the binary SVM classifier to the multi-class problem. Specifically, we looked into the margins that are being maximized by existing methods: 1-vs-1, 1-vs-Rest, and the two unified variants of 1-vs-Rest – C&S (Crammer & Singer, 2001) and W&W (Weston & Watkins, 1998). We started from the observation that the margins that the methods aim to maximize are not the margins with respect to the decision boundaries of the final classifier. This seems to undermine the viability of these methods. However, our main result indicates that these methods effectively maximize the margins with respect to the decision boundaries as one would hope to do. Another result shows that the W&W variant is identical to the 1-vs-1 classifier when the latter is constrained so that each binary classifier is the difference between two uni-class separators. These results support empirical evidence from the literature showing that different multi-class SVM classifiers perform well and rather similarly, as we show in our experiments as well.

Visual Object Recognition with Sparse Features Derived from a Convolution Neural Net

Abstract

Recent multi-class visual object recognition studies, favor 1-vs-Rest SVM with descriptors extracted from pre-trained CNNs when the image database contains many different classes. In our work, we address the sparsity of such descriptors. With CNN features, we observed that for 1-vs-Rest, the use of binary descriptors (by quantizing the CNN features) yields comparable results to utilizing the full feature value. Moreover, for Nearest Neighbor and Image Retrieval, the binary descriptors improve classification results.

1. Introduction

Deep convolution neural networks currently achieve state-of-the-art classification results on image classification tasks (Krizhevsky et al., 2012); the main drawback of Deep CNNs is that, due to their large number of parameters, they require abundant (thousands) training samples in order to be trained effectively.

Upon studying the Deep CNNs layers it was concluded that deep layers can form image descriptors that represent the image better than handmade features (Zeiler and Fergus 2013). Thus a new method emerged: utilizing descriptors extracted from pre-trained Deep CNNs and building a new classifier on top of those descriptors that classifies according to the new classes. Razavian et al. (2004) and Donahue et al. (2014) revealed that this method yields cutting-edge results and can be used even with a relatively small training set (a few dozen samples from each class).

Interestingly, in the description layer (one layer before the classification layer) the descriptor tends to be sparse, this is mostly a result of the ReLU (used inside the neuron), which zeros out all negative values.

In section 2 we will look at the quantization of the descriptor. In section 3 we will explore a new classification method using quantized data.

2. Quantizing the CNN features

In this section we present an empirical observation on the Caltech 256 (Griffin et al. 2006) and PASCAL VOC 2007 (Everingham et al. 2012) databases.

2.1 Method

Images are re-sized so that their small edge is 231, then the center is cropped to 231x231 and fed to the pre-trained CNN Overfeat (Sermanet et al. 2014 - using the small, 'fast' version). The features used are taken from the 20th layer (right before the soft-max), resulting in a 4096 feature vector. The vector is then normalized in each dimension so that the features are in [0,1].

In preliminary tests, we attempted to use the data from layer 19 – before the ReLU but this produced less favorable results.

Train: in additional to the train images we added the images mirrored horizontally (mirroring has a significant effect on the features as suggested by Zeiler and Fergus, 2013).

2.2 DataSets

Caltech 256 DB: Contains 256 classes, each class has at least 80 samples. When testing this dataset we chose 5 random 60/20 train-test splits and reported the results on the test part.

PASCAL VOC 2007: Contains 20 classes. The objects are not centered. When testing this dataset we used the given train/test split, all images with more than one label or marked as "problematic labels" were removed. Bounding box annotations were not used.

2.3 SVM-parameters

We used LibLinear (Fan et al., 2008) for 1-vs-Rest-SVM and Crammer & Singer. A 3-folod cross validation is used to tune the C parameter, our first scan was $C = 2^{-13}, 2^{-12} \dots 2^{3}$ we then performed a denser scan of ± 0.5 (of the power) near the best result, and repeated the process for 0.25 (following the paradigm of Hsu et al. 2003, but discluding higher C values as the produced values were smaller than 2^{1}).

For Nearest Neighbor we used the MATLAB implementation.

2.4 Results

Empirical observations: The representation features learned by the pre-trained CNN, are sparse, 86% of the features are 0 (on both databases).

TABLE 1
TEST ERROR QUANTIZATION RESULTS

Caltech 256	No Quantization	0/1
1-vs-Rest	34.63±1.0	34.70±0.8
Crammer & Singer	33.91±0.6	33.18±0.7
Nearest Neighbor	53.73±0.3	47.66±0.4
VOC 2007	No Quantization	0/1
1-vs-Rest	20.54	20.36
Crammer & Singer	21.18	20.01
Nearest Neighbor	34.44	27.22

* For all methods listed above, train error is zero.

As seen in Table 1, with 1-vs-Rest or Crammer & Singer, the test results are near identical. When using Nearest Neighbor, quantization of the feature vectors provides superior test results.

3. Improved Nearest Neighbor

We created a class descriptor for each training class, it is a feature vector that has 1 if at least half the class’ samples have 1 and 0 otherwise (it represents the class’ median). We used the class' descriptors as training samples to which we applied either Nearest Neighbor (“Median-NN”) or Crammer & Singer (“Median-C&S”).

TABLE 2
0/1 data Classification Methods

Caltech 256	Test Error	Train Error
Nearest Neighbor	47.66±0.40	0
Median-NN	42.84±0.58	37.52±0.25
Crammer & Singer	33.18±0.70	0
Median-C&S	55.37±0.43	51.71±1.31

The results in Table 2 reveal that although the train Error of Median-NN is larger, its test Error is better than regular NN. For C&S the median descriptors yields less favorable results than C&S.

4. Image Retrieval

Given an image query, the method we suggest, returns the K closest images from the database (according to Euclidean distance in the feature space). In these experiments the query images are the test images from the classification experiments. The train images are the database. An image is considered a match if it is from the same class as the query image.

TABLE 3
Image Retrieval Quantization Error results

1 Image Retrieval	No Quantization	0/1
Caltech 256 - CNN	55.66±0.7	48.45±0.3
VOC 2007 - CNN	34.44	27.22

20 Image Retrieval	No Quantization	0/1
Caltech 256 - CNN	74.18±0.4	66.62±0.2
Caltech 256 - Yang et al., (2014)	91	-

As seen in Table 3, quantization of the feature vectors provides superior image retrieval results. We tried to replicate the experiment of Yang et al., (2014) by retrieving the 20 closest images. Using our method with quantization achieved 24% accuracy improvement.

5. Summary

In this chapter we studied the effect of quantizing the features extracted from a pre trained CNN. We observed that for 1-vs-Rest and Crammer & Singer the quantization yields comparable results to those found when using the full feature value. We then tested Nearest Neighbor (NN), observing that the quantization improved classification. Our next step was to take advantage of this fact, creating Median-NN, which is more efficient than NN, and results in better classification accuracy (although still incomparable to 1-vs-Rest SVM).

Tree Classification and Novelty Detection Using Features Derived from a Convolution Neural Net

Abstract

Recent studies of multi-class visual object recognition, with a large database of images containing a multitude of various objects, favor using a classification method built on top of descriptors extracted from pre-trained CNNs. In this chapter we examine hierarchical tree-like meta-structures, which describe the set of classes, discovering that the learned classification trees resemble those reported by human observers. We sought to utilize these trees for information transfer to new classes of object, where the task is to recognize that a sample is novel, and use the tree to bootstrap the classification of the new classes. Our current methods can only determine novelty of groups of samples, whereas 1-vs-Rest SVM produces favorable results for retraining. We present empirical results on 27 classes of the Caltech256 image dataset.

1. Introduction

The motivation for the use of a meta-class structure is twofold; the first is complexity; in regards to the number of classes, "flat" multi-class classification strategies, such as 1-vs-Rest, have linear test complexity in the number of classes, while a tree-like structure can achieve logarithmic test complexity. The second motivation is novelty detection and information transfer.

When learning in a setting numerous classes, it is crucial that the meta-structure is built automatically.

Related work:

Some studies present methods for automatic tree creation. Platt et al. (2000) created tree shaped DAGs for which one class is disqualified at each level. Gao & Koller (2011) created a tree by separating the classes into three groups at each node and creating a classifier that separates two of the groups and ignores the third one.

Other researches utilized a handmade given hierarchy tree and examined novelty detection and information transfer such as Dekel et al. (2004),Weinshall et al. (2008), Rohrbach et al. (2011), and Coppi et al. (2014).

Other researchers explored both problems combined: Liu et al. (2011) looked for the two groups of classes that can be separated with the biggest SVM margin for each node (using the approximation of the Constrained Concave-Convex procedure). Fan et al. (2014), used the similarity of various SIFT and GIST statistics to create a classification tree.

Bodesheim et al. (2015) used local learning for multiclass novelty detection (without trees).

2. The Dataset

We used a subset of classes from the Caltech 256 image database. As in the previous section, all images were re-sized to 231x231 and fed to the pre-trained CNN Overfeat. The features used were taken from the 20 layer (immediately prior to the soft-max), resulting in a 4096 feature vector, (similarly to the past section, for this section, in preliminary tests, we attempted to use the layer 19 data – prior to the ReLU but obtained less favorable results; using 0/1 values instead of the full values, gave comparable results).

Train: for all 60 train images we also added the images mirrored horizontally.

Test: 20 images (no mirroring).

- SimpleTrainTest – the Train samples are the first 60 images of each class, the Test images are the following 20 (for reproducibility).
- 27 handpicked classes: these classes were handpicked, below is an example of a handmade tree of these classes (in our experiments, five subjects were asked to create trees out of the class labels).

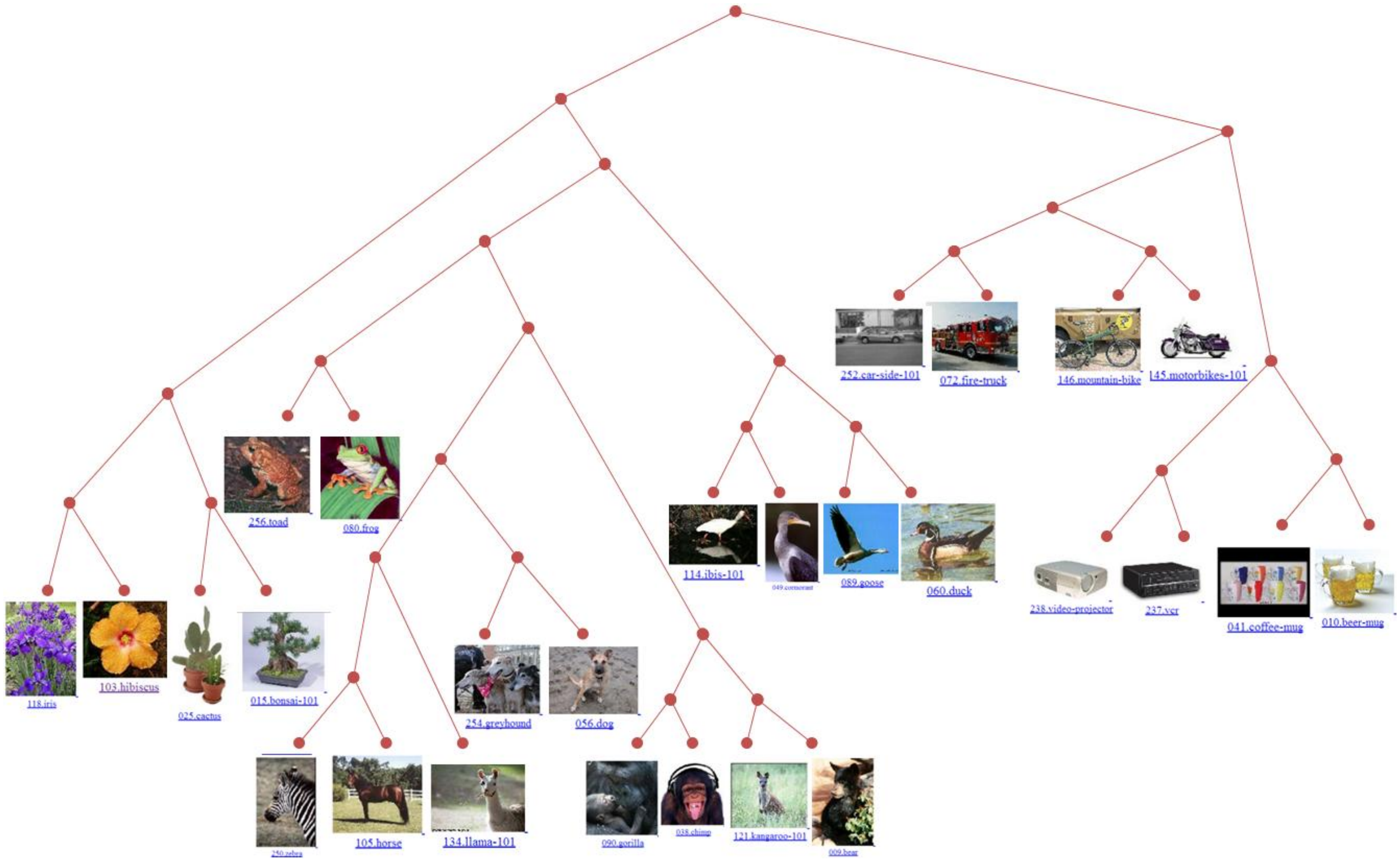


Fig. 1. 27 Handmade Tree

3. Methods

During our research, we examined four main binary tree building paradigms.

3.1 GMM-SVM-Tree

We expect this method to be the most scalable.

For each tree node we applied Constrained-GMM in order to split the classes into two groups (Positively-Constrained-GMM is equivalent to GMM on the class' centers as proved by Shental et al. 2003). Then we used binary-SVM on all node samples (in accordance with the GMM created groups). The train/test samples continued on to the next node in accordance with the aforementioned SVM classifier. If a node has only one class, it is defined as a leaf with the class' label.

(In practice, for each node, we dropped classes that possessed less than 3 samples. We conducted 5 runs of the GMM-SVM-Tree and selected the most appropriate classifier using the SVM's 3-fold cross validation score, for each run. We conducted 10 runs of the GMM and selected the one that was most balanced).

For the SimpleTrainTest 27 classes, GMM-SVM-Tree produced slightly varied tree structures for different runs. All the GMM-SVM-Trees that we observed maintained most of the human created clusters; the following is an example of a "bad" tree:

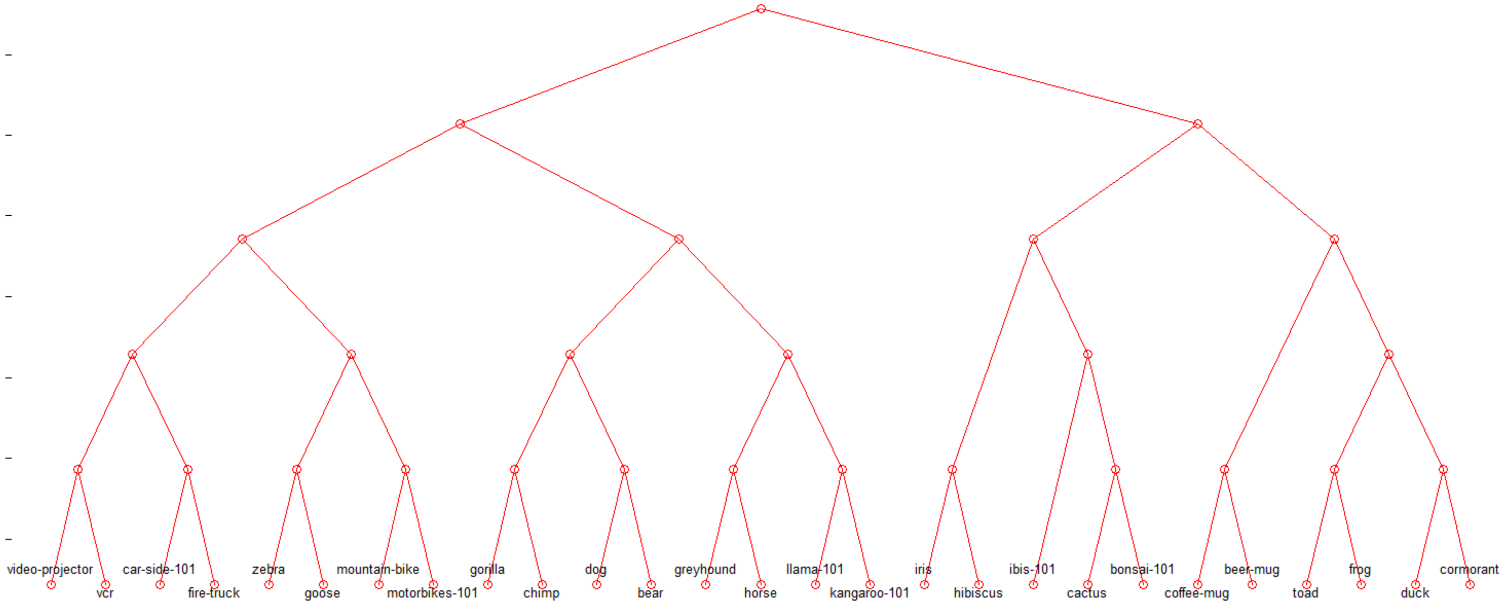


Fig. 2. "bad" GMM-SVM-Tree, 27 SimpleTrainTest Test Error: 16.3%, Train Error: 0.03%
* This tree is on 0/1 data, regular data gave comparable results.

As is evident in fig. 2, some human defined clusters remain but there are several "outliers" (such as the ibis-101 that joins the flowers), most of the pairs remain logical.

The following is an example of a "good" tree:

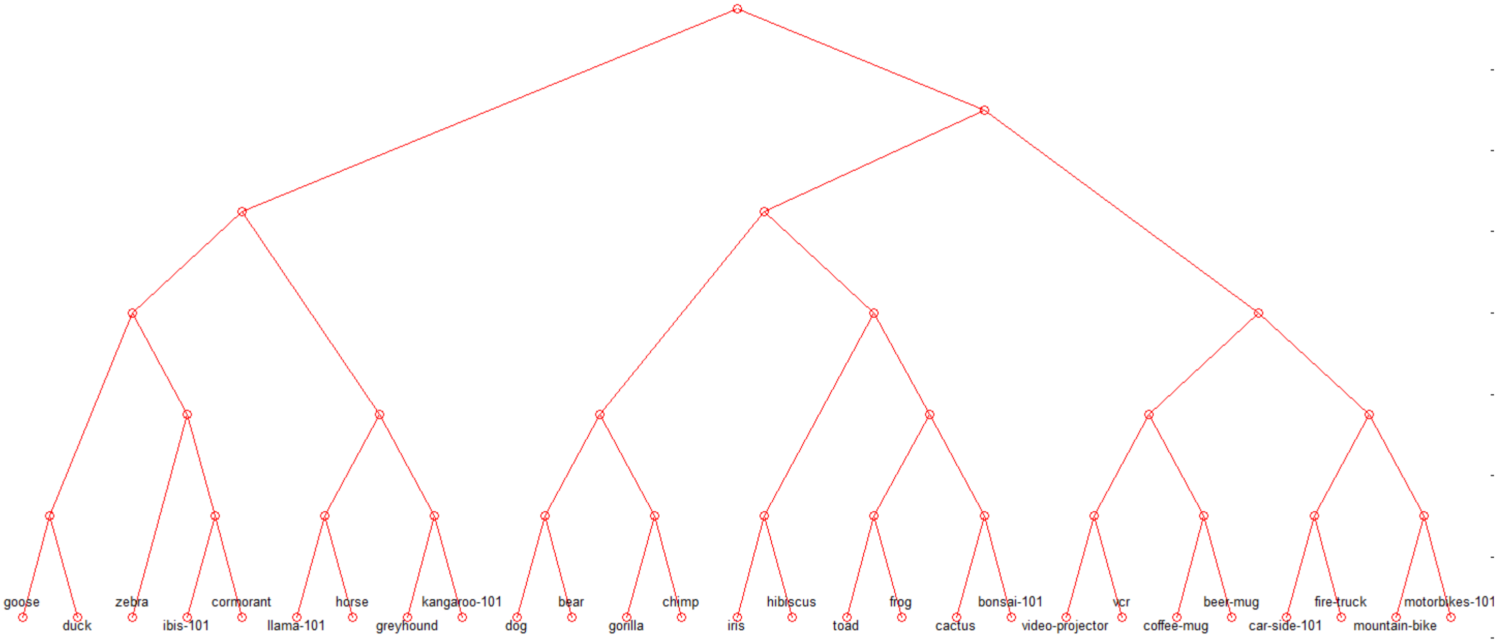
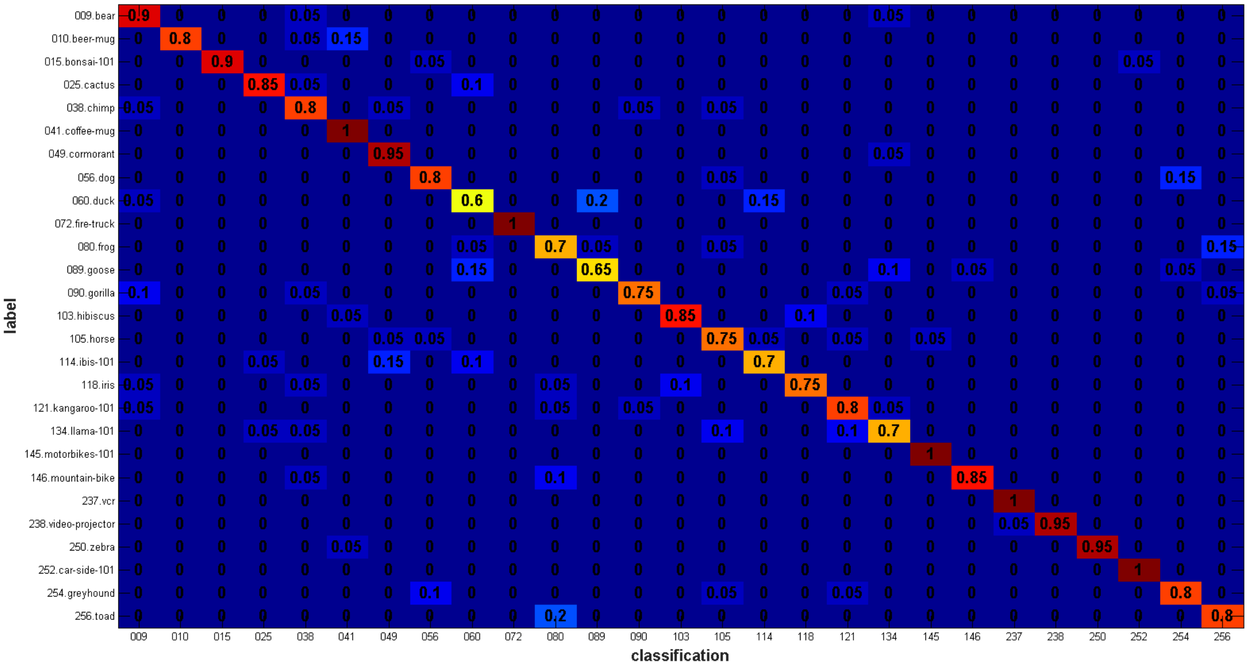


Fig. 3. "good" GMM-SVM-Tree, 27 SimpleTrainTest Test Error: 14.8%, Train Error: 0%

* This tree is on 0/1 data, regular data gave comparable results.

Fig.3 contains multiple human perceived clusters such as "manmade" to the right or "birds" (with zebra as an outlier). Both "bad" and "good" trees produce similar confusion matrix. The "bad" tree confusion matrix is presented here:



	009	010	015	025	038	041	049	056	060	072	080	089	090	103	105	114	118	121	134	145	146	237	238	250	252	254	256
009.bear	0.9	0	0	0	0.05	0	0	0	0	0	0	0	0	0	0	0	0	0	0.05	0	0	0	0	0	0	0	0
010.beer-mug	0	0.8	0	0	0.05	0.15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
015.bonsai-101	0	0	0.9	0	0	0	0	0.05	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.05	0	0
025.cactus	0	0	0	0.85	0.05	0	0	0	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
038.chimp	0.05	0	0	0	0.8	0	0.05	0	0	0	0	0	0.05	0	0.05	0	0	0	0	0	0	0	0	0	0	0	0
041.coffee-mug	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
049.cormorant	0	0	0	0	0	0	0.95	0	0	0	0	0	0	0	0	0	0	0	0.05	0	0	0	0	0	0	0	0
056.dog	0	0	0	0	0	0	0	0.8	0	0	0	0	0	0	0.05	0	0	0	0	0	0	0	0	0	0	0.15	0
060.duck	0.05	0	0	0	0	0	0	0	0.6	0	0	0.2	0	0	0	0.15	0	0	0	0	0	0	0	0	0	0	0
072.fire-truck	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
080.frog	0	0	0	0	0	0	0	0	0.05	0	0.7	0.05	0	0	0.05	0	0	0	0	0	0	0	0	0	0	0	0.15
089.goose	0	0	0	0	0	0	0	0	0.15	0	0	0.65	0	0	0	0	0	0	0.1	0	0.05	0	0	0	0	0.05	0
090.gorilla	0.1	0	0	0	0.05	0	0	0	0	0	0	0	0.75	0	0	0	0	0.05	0	0	0	0	0	0	0	0	0.05
103.hibiscus	0	0	0	0	0	0.05	0	0	0	0	0	0	0	0.85	0	0	0.1	0	0	0	0	0	0	0	0	0	0
105.horse	0	0	0	0	0	0	0.05	0.05	0	0	0	0	0	0	0.75	0.05	0	0.05	0	0.05	0	0	0	0	0	0	0
114.ibis-101	0	0	0	0.05	0	0	0.15	0	0.1	0	0	0	0	0	0	0.7	0	0	0	0	0	0	0	0	0	0	0
118.iris	0.05	0	0	0	0.05	0	0	0	0	0	0.05	0	0	0.1	0	0	0.75	0	0	0	0	0	0	0	0	0	0
121.kangaroo-101	0.05	0	0	0	0	0	0	0	0	0	0.05	0	0.05	0	0	0	0	0.8	0.05	0	0	0	0	0	0	0	0
134.llama-101	0	0	0	0.05	0.05	0	0	0	0	0	0	0	0	0	0.1	0	0	0.1	0.7	0	0	0	0	0	0	0	0
145.motorbikes-101	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
146.mountain-bike	0	0	0	0	0.05	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	0	0.85	0	0	0	0	0	0
237.vcr	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
238.video-projector	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.05	0.95	0	0	0	0
250.zebra	0	0	0	0	0	0.05	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.95	0	0	0
252.car-side-101	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
254.greyhound	0	0	0	0	0	0	0	0.1	0	0	0	0	0	0	0.05	0	0	0.05	0	0	0	0	0	0	0	0.8	0
256.toad	0	0	0	0	0	0	0	0	0	0	0.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.8

Fig. 4. "bad" GMM-SVM-Tree Confusion matrix, 27 SimpleTrainTest

It is evident from the confusion matrix in fig. 4 that the large misclassification errors are performed for pairs of classes that are close in the tree and in the human perception (toad-frog, ibis-duck, dog-greyhound, beer-coffee mug), however, some of the existing errors still exist in unrelated classes (e.g. chimp-cormorant).

3.2 TSVM-SVM-Tree

This method is motivated by the hypothesis that it will benefit the overall performance if the clustering performed at each level is optimized (at least partially) to achieve good classification for the emerging clusters. For this reason we chose the transductive SVM method (Gammerman & Vapnik, 1998), which attempts to maximize the unsigned margin of the unlabeled points as well.

Specifically, we started by computing the centers of all classes, seeking TSVM-based clustering for those points only. We selected two classes at random for each node, then ran Transductive-SVM (using SVMlight, Joachims 1999) on the class' centers (which identifies the line that separates the centers and separates the two chosen classes into different groups, maximizing the margin from any other class center). Then as in GMM, an SVM classifier was trained on all samples. (In practice, as in GMM-SVM, we dropped smaller classes; we selected two random classes for each run and trained TSVM with balance value of steps from 30% to 70%). The TSVM-SVM tree was implemented by Reuven Siman-Tov .

Working with centers - SVM Dual Perspective:

Using the classes centers requires significantly fewer calculations. In this section we will show that it makes theoretical sense.

Let $S = \{(\bar{x}_{1,1}, y_1), \dots, (\bar{x}_{1,\tilde{m}}, y_1), \dots, (\bar{x}_{k,1}, y_k), \dots, (\bar{x}_{k,\tilde{m}}, y_k)\}$ denote a set of m training examples, indexed by their classes, where $\bar{x}_{i,j} \in \mathbb{R}^n$ and labels $y_i \in \{-1,1\}$ / Define: $\tilde{x}_i = \frac{1}{\tilde{m}} \sum_{j=1}^{\tilde{m}} \bar{x}_{i,j}$ the class' centers.

Definition 1: binary SVM – on class' centers

$$H(\bar{x}) = sign(\bar{w}^T \bar{x})$$

Where $\bar{w} \in \mathbb{R}^n$ are obtained by solving:

$$\min_{w,\varepsilon_i\geq 0} \ \frac{1}{2}\|w\|^2 + \tilde{C}\sum_{i=1}^{k}\tilde{\varepsilon}_i$$

$$s.t. \forall\, i \in [k] \quad y_i * w * \tilde{x}_i \geq 1 - \tilde{\varepsilon}_i$$

<u>Binary SVM – on class' centers - Dual</u>:

$$\max_{\tilde{\alpha}_i,\beta_i\geq 0} min_{w,\tilde{\varepsilon}_i} \frac{1}{2}\|w\|^2 + \tilde{C}\textstyle\sum_{i=1}^{k}\tilde{\varepsilon}_i - \sum_{i=1}^{k}\tilde{\alpha}_i(y_i * w * \tilde{x}_i - 1 + \tilde{\varepsilon}_i) - \sum_{i=1}^{k}\beta_i\tilde{\varepsilon}_i$$

w is unconstrained, therefore, in any extremal point of this function, its derivative with respect to w must be 0. Since the second derivative with respect to w is positive, this extremum is a minimum. Hence, we can derive the value of w at a minimum:

$$0 = \frac{\partial F}{\partial w} = w - \sum_{i=1}^{k}\tilde{\alpha}_i(y_i * \tilde{x}_i) \ \Rightarrow \quad w = \sum_{i=1}^{k} y_i\tilde{\alpha}_i\tilde{x}_i$$

$\tilde{\varepsilon}_i$ can lead to $-\infty$ therefore its multiplayer must be 0:

$$\forall\, i \in [k] \quad \tilde{C} - \tilde{\alpha}_i - \beta_i = 0$$

The dual optimization problem is equal to:

$$\max_{\tilde{\alpha}_i,\beta_i\geq 0} \frac{1}{2}\|w\|^2 - \sum_{i=1}^{k}\tilde{\alpha}_i(y_i * w * \tilde{x}_i - 1)$$

$$s.t. \qquad \forall\, i \in [k] \quad \tilde{C} - \tilde{\alpha}_i - \beta_i = 0 \ \ , w = \sum_{i=1}^{k} y_i\tilde{\alpha}_i\tilde{x}_i$$

β_i is positive and appears only at the constraint, so the constraint can be replaced by: $\tilde{C} \geq \tilde{\alpha}_i$, arranging the dual well lead to:

$$\max_{\tilde{\alpha}_i\geq 0} \frac{1}{2}\|w\|^2 - \sum_{i=1}^{k}\tilde{\alpha}_i(y_i * w * \tilde{x}_i - 1)$$

$$s.t. \qquad \forall\, i \in [k] \quad \tilde{C} \geq \tilde{\alpha}_i \ , \qquad w = \sum_{i=1}^{k} y_i\tilde{\alpha}_i\tilde{x}_i$$

With $\tilde{x}_i = \frac{1}{\tilde{m}}\sum_{j=1}^{\tilde{m}} \bar{x}_{i,j}$, we will add a change of variables: $\tilde{\alpha}_i = \alpha_i\tilde{m}$, $\tilde{C} = C\tilde{m}$

$$\max_{\alpha_i \geq 0} \frac{1}{2}\|w\|^2 - \sum_{i=1}^{k} \alpha_i \widetilde{m}(y_i * w * \tilde{x}_i - 1)$$

$$s.t. \qquad \forall\, i \in [k] \quad C \geq \alpha_i \ , \qquad w = \widetilde{m}\sum_{i=1}^{k} y_i \alpha_i \tilde{x}_i$$

Therefore:

$$H(\bar{x}) = sign(\widehat{w}^T \bar{x}) = sign\left(\sum_{i=1}^{k} y_i \alpha_i \sum_{j=1}^{\widetilde{m}} \bar{x}_{i,j}{}^T \bar{x}\right)$$

In the same manner, for binary SVM (on all points), we will get:

$$H(\bar{x}) = sign\left(\sum_{i=1}^{k} y_i \sum_{j=1}^{\widetilde{m}} \alpha_{i,j}\, \bar{x}_{i,j}{}^T \bar{x}\right)$$

Conclusion: Using the class' centers as the dots for the node's classifier, is identical to adding the constraint: $\alpha_{i,j} = \alpha_i$.

$\bar{x}_{i,j}{}^T \bar{x}$ is perceived as a similarity factor between $\bar{x}$ and $\bar{x}_{i,j}$. Therefore, $\alpha_{i,j}$ serve as importance factors (if $\alpha_{i,j}$ has a high value, and $\bar{x}$, $\bar{x}_{i,j}$ are similar, the label of $\bar{x}$ should be the same as $\bar{x}_{i,j}$).

Therefore, using the class' centers is equal to maintaining the same importance factor for all samples from the same class. This makes theoretical sense, since the class' samples should have the same ± 1 label.

Note: since Chapelle et al. (2008) questioned SVMlight's convergence to a global minimum, we also implemented a branch and bound function that detects the global minimum of Tsvm. It didn't improve the TSVM-SVM-Tree results on our 27 classes.

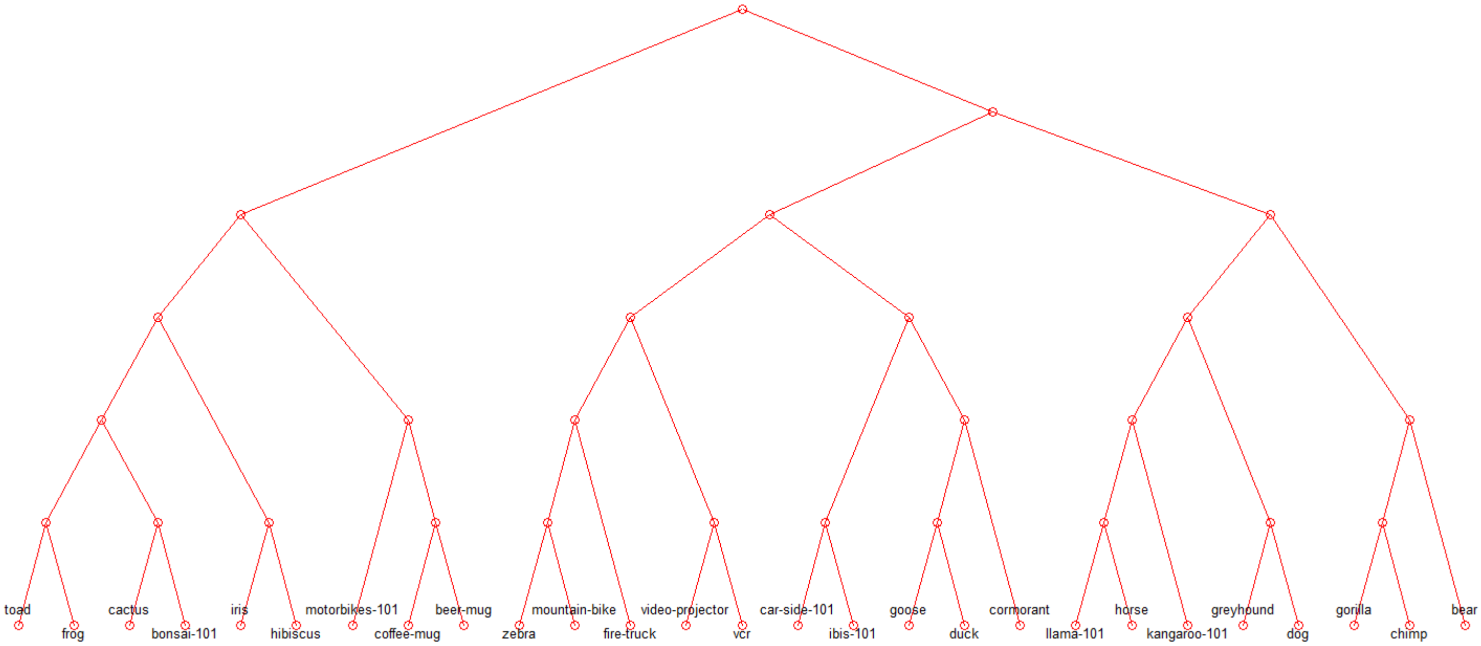
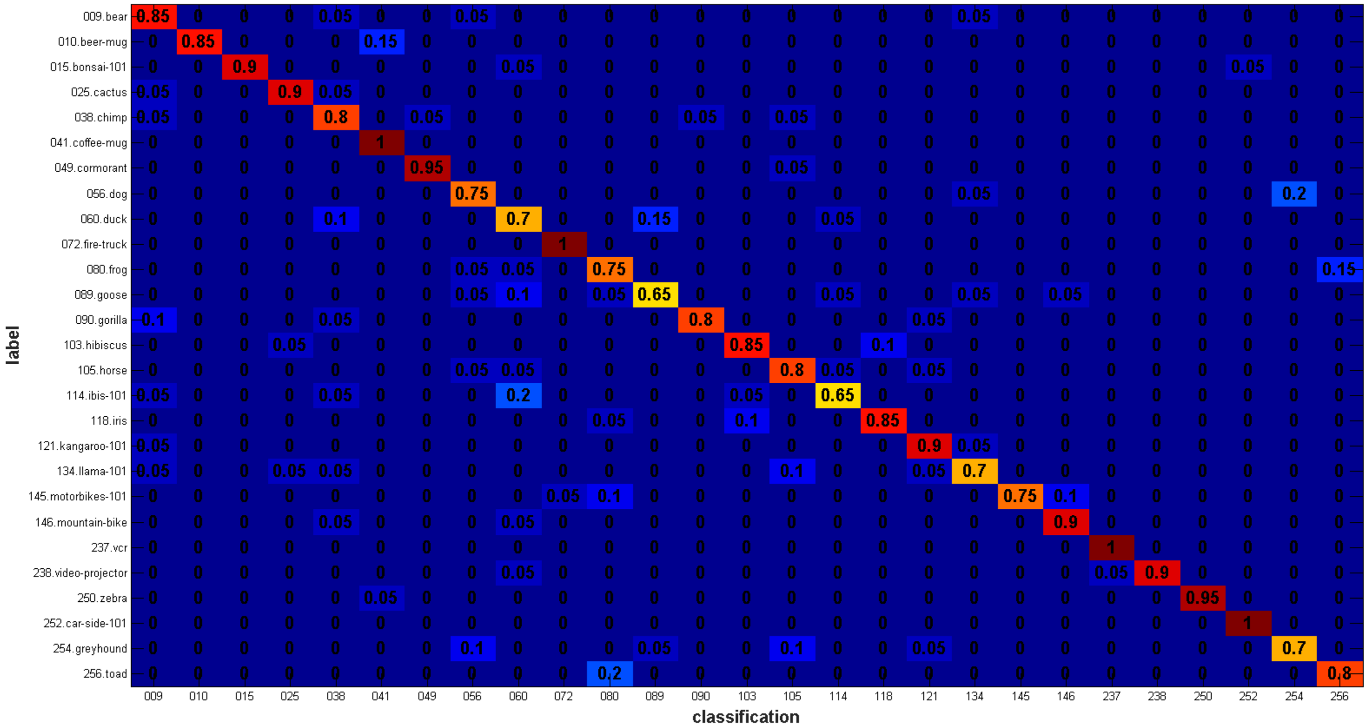


Fig. 5. TSVM-SVM-Tree, 27 SimpleTrainTest Test Error: 15±1%, Train Error: 0
* This tree is on 0/1 data, regular data gave comparable results.

Fig. 5 presents one of the TSVM-SVM-Trees. For multiple runs (still SimpleTrainTest) the tree maintains a similar form, precluding the outliers, such as zebra or car-side-101 that tended to switch places).



	009	010	015	025	038	041	049	056	060	072	080	089	090	103	105	114	118	121	134	145	146	237	238	250	252	254	256
009.bear	0.85	0	0	0	0.05	0	0	0.05	0	0	0	0	0	0	0	0	0	0	0.05	0	0	0	0	0	0	0	0
010.beer-mug	0	0.85	0	0	0	0.15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
015.bonsai-101	0	0	0.9	0	0	0	0	0	0.05	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.05	0	0
025.cactus	0.05	0	0	0.9	0.05	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
038.chimp	0.05	0	0	0	0.8	0	0.05	0	0	0	0	0	0.05	0	0.05	0	0	0	0	0	0	0	0	0	0	0	0
041.coffee-mug	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
049.cormorant	0	0	0	0	0	0	0.95	0	0	0	0	0	0	0	0.05	0	0	0	0	0	0	0	0	0	0	0	0
056.dog	0	0	0	0	0	0	0	0.75	0	0	0	0	0	0	0	0	0	0	0.05	0	0	0	0	0	0	0.2	0
060.duck	0	0	0	0	0.1	0	0	0	0.7	0	0	0.15	0	0	0	0.05	0	0	0	0	0	0	0	0	0	0	0
072.fire-truck	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
080.frog	0	0	0	0	0	0	0	0.05	0.05	0	0.75	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.15
089.goose	0	0	0	0	0	0	0	0.05	0.1	0	0.05	0.65	0	0	0	0.05	0	0	0.05	0	0.05	0	0	0	0	0	0
090.gorilla	0.1	0	0	0	0.05	0	0	0	0	0	0	0	0.8	0	0	0	0	0.05	0	0	0	0	0	0	0	0	0
103.hibiscus	0	0	0	0.05	0	0	0	0	0	0	0	0	0	0.85	0	0	0.1	0	0	0	0	0	0	0	0	0	0
105.horse	0	0	0	0	0	0	0	0.05	0.05	0	0	0	0	0	0.8	0.05	0	0.05	0	0	0	0	0	0	0	0	0
114.ibis-101	0.05	0	0	0	0.05	0	0	0	0.2	0	0	0	0	0.05	0	0.65	0	0	0	0	0	0	0	0	0	0	0
118.iris	0	0	0	0	0	0	0	0	0	0	0.05	0	0	0.1	0	0	0.85	0	0	0	0	0	0	0	0	0	0
121.kangaroo-101	0.05	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.9	0.05	0	0	0	0	0	0	0	0
134.llama-101	0.05	0	0	0.05	0.05	0	0	0	0	0	0	0	0	0	0.1	0	0	0.05	0.7	0	0	0	0	0	0	0	0
145.motorbikes-101	0	0	0	0	0	0	0	0	0	0.05	0.1	0	0	0	0	0	0	0	0	0.75	0.1	0	0	0	0	0	0
146.mountain-bike	0	0	0	0	0.05	0	0	0	0.05	0	0	0	0	0	0	0	0	0	0	0	0.9	0	0	0	0	0	0
237.vcr	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
238.video-projector	0	0	0	0	0	0	0	0	0.05	0	0	0	0	0	0	0	0	0	0	0	0	0.05	0.9	0	0	0	0
250.zebra	0	0	0	0	0	0.05	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.95	0	0	0
252.car-side-101	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
254.greyhound	0	0	0	0	0	0	0	0.1	0	0	0	0.05	0	0	0.1	0	0	0.05	0	0	0	0	0	0	0	0.7	0
256.toad	0	0	0	0	0	0	0	0	0	0	0.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.8

Fig. 6. TSVM-SVM-Tree Confusion matrix, 27 SimpleTrainTest

In fig. 6, we see that similarly to the GMM-SVM-Tree, the big misclassification errors occur for pairs of classes that are close in the tree, and in human perception (toad-frog, ibis-duck, dog-greyhound).

3.3 C-Kmeans-SVM-Tree

This method does not solely examine centers in order to split the classes; instead, it surveys all given samples and uses their class' label as an indication for a positive constraint.

For each node, we used Constrained-Kmeans (Wagstaff et al. 2001) (Hu et al., 2008), K-means which prefers to keep samples from the same class together. We then applied SVM according to the labels (In practice as in GMM-SVM we dropped smaller classes and conducted 5 runs).

C-Kmeans-SVM-Tree provides comparable results to TSVM-SVM-Tree on the 27 SimpleTrainTest.

3.4 Two-Split-Tree

This method creates large trees. As seen in the GMM-SVM-Tree, some of the errors occur in unrelated classes; by transferring several of the same class to both sides of the node and allowing them to create classification leaves on both sides, we hoped to solve this problem.

First, we trained two 1-vs-1 separators for each pair of classes, each separator hyper plane was selected to be close to one of the groups (this was performed by setting a different SVM error weight C for each class; the two weights were found using a binary search). Then, at each node, we selected the two classes that resulted in minimal class movement to both sides. The test separator of the node was the subtraction of the "close to class" separators (see fig. 7). In training, a class can go to one side of the node (as in fig 7). Else it will travel to both sides.

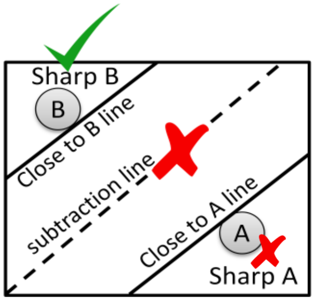
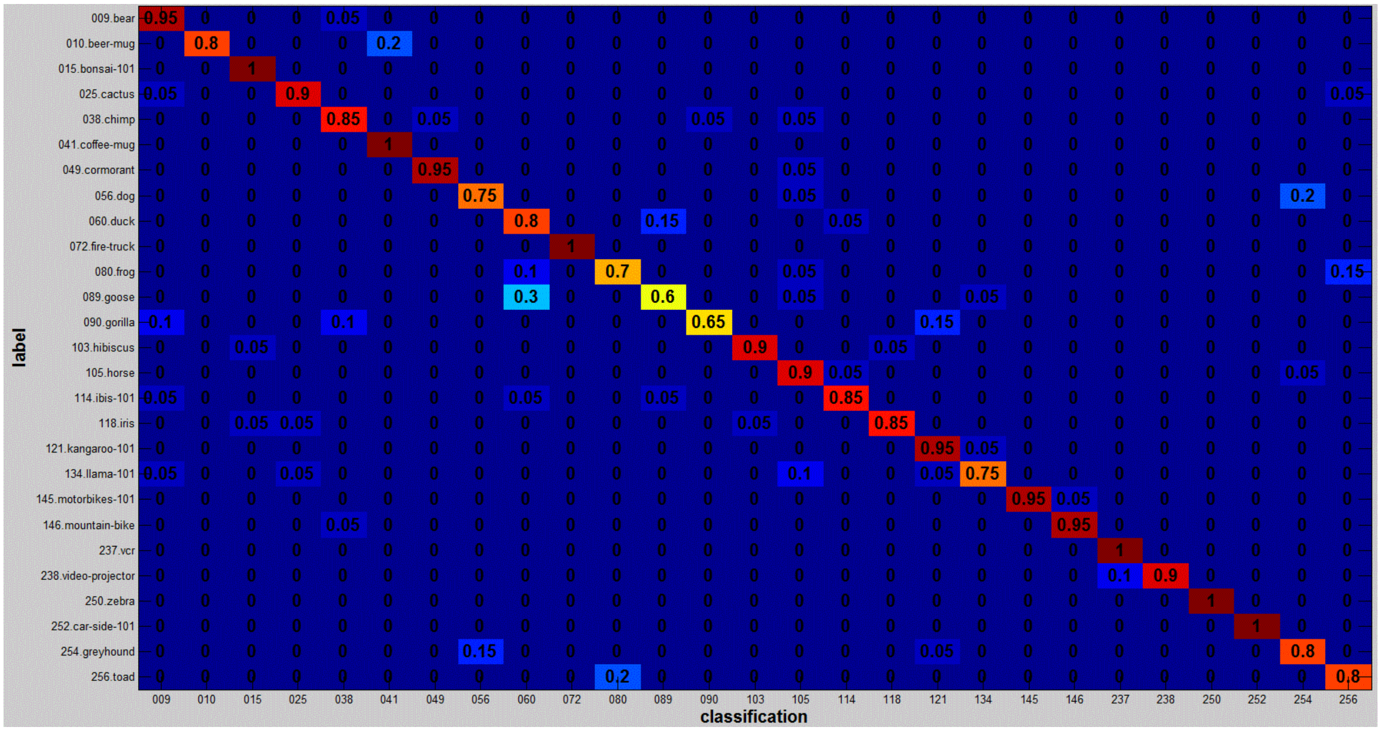


Fig. 7. The class will travel to side B if:

✓ = There is a considerable amount of samples in this Area (10 or more).
✗ = There are less than 10 samples in these Areas.

The two classes in each node are classes that made the least amount of movements between both sides.



	009	010	015	025	038	041	049	056	060	072	080	089	090	103	105	114	118	121	134	145	146	237	238	250	252	254	256
009.bear	0.95	0	0	0	0.05	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
010.beer-mug	0	0.8	0	0	0	0.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
015.bonsai-101	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
025.cactus	0.05	0	0	0.9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.05
038.chimp	0	0	0	0	0.85	0	0.05	0	0	0	0	0	0.05	0	0.05	0	0	0	0	0	0	0	0	0	0	0	0
041.coffee-mug	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
049.cormorant	0	0	0	0	0	0	0.95	0	0	0	0	0	0	0	0.05	0	0	0	0	0	0	0	0	0	0	0	0
056.dog	0	0	0	0	0	0	0	0.75	0	0	0	0	0	0	0.05	0	0	0	0	0	0	0	0	0	0	0.2	0
060.duck	0	0	0	0	0	0	0	0	0.8	0	0	0.15	0	0	0	0.05	0	0	0	0	0	0	0	0	0	0	0
072.fire-truck	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
080.frog	0	0	0	0	0	0	0	0	0.1	0	0.7	0	0	0	0.05	0	0	0	0	0	0	0	0	0	0	0	0.15
089.goose	0	0	0	0	0	0	0	0	0.3	0	0	0.6	0	0	0.05	0	0	0	0.05	0	0	0	0	0	0	0	0
090.gorilla	0.1	0	0	0	0.1	0	0	0	0	0	0	0	0.65	0	0	0	0	0.15	0	0	0	0	0	0	0	0	0
103.hibiscus	0	0	0.05	0	0	0	0	0	0	0	0	0	0	0.9	0	0	0.05	0	0	0	0	0	0	0	0	0	0
105.horse	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.9	0.05	0	0	0	0	0	0	0	0	0	0.05	0
114.ibis-101	0.05	0	0	0	0	0	0	0	0.05	0	0	0.05	0	0	0	0.85	0	0	0	0	0	0	0	0	0	0	0
118.iris	0	0	0.05	0.05	0	0	0	0	0	0	0	0	0	0.05	0	0	0.85	0	0	0	0	0	0	0	0	0	0
121.kangaroo-101	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.95	0.05	0	0	0	0	0	0	0	0
134.llama-101	0.05	0	0	0.05	0	0	0	0	0	0	0	0	0	0	0.1	0	0	0.05	0.75	0	0	0	0	0	0	0	0
145.motorbikes-101	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.95	0.05	0	0	0	0	0	0
146.mountain-bike	0	0	0	0	0.05	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.95	0	0	0	0	0	0
237.vcr	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
238.video-projector	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1	0.9	0	0	0	0
250.zebra	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
252.car-side-101	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
254.greyhound	0	0	0	0	0	0	0	0.15	0	0	0	0	0	0	0	0	0	0.05	0	0	0	0	0	0	0	0.8	0
256.toad	0	0	0	0	0	0	0	0	0	0	0.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.8

Fig. 8. Two-Split-Tree Confusion matrix, 27 SimpleTrainTest Test Error: 12.77±2%, Train Error: 1.1%

While the trees produced had relatively good test error, they had many more nodes (63,000 leaves compared to the 27 leaves of the others). Their training time was almost triple, as seen from fig 8. Several unrelated misclassifications still occurred (llama- cactus).

3.5 Results

We present the results of some of the methods on the 27 classes handpicked from the Caltech 256 dataset.

TABLE 1
5 train/test split results

Data*	**Caltech 27****	Test Error	Train Error	Tree Height	Leaves num
	Crammer & Singer	13.37±1.15	0	-	-
0/1	*Crammer & Singer*	13.33±0.74	0.01±0.05	-	-
	GMM-SVM-Tree	15.44±1.04	0.04±0.06	6	27
0/1	*GMM-SVM- Tree*	15.41±1.15	0.04±0.04	6	27
	*Two-Split-Tree****	14.70±1.93	1.24±0.64	24-26	63,000-99,000
0/1	*Two-Split-Tree****	15.55±1.85	3.12±0.71	23-25	28,000-64,000

* 0/1 is the data quantized so that all values that are not 0 are 1
** 27 are hand-picked classes of the Caltech 256.
*** We tested a different threshold for the Two-Split-Tree; using 7 instead of 10 produced trees with double the amount of leaves, but comparable classification error. Whereas, using 30 produced smaller trees but increased classification error by more than 4%.

As seen from table 1 Crammer & Singer slightly outperforms both trees in classification error.

4. Novelty detection

Our goal was to not only recognize that this is a novel class (a class that wasn't evident in the train phase), but also to detect the part of the tree in which it is supposed to exist (described in 4.1) and later to retrain part of the tree to classify the new class (section 4.2).

4.1 Meta class novelty detection

We aim to detect a novel class and determine the part of the tree to which the class belongs. We began with 27 classes, and removed 3 classes: the goose (which was expected to be classified with the birds), the beer mug (which is close to the coffee mug) and the VCR (paired with the video projector).

Method: when a sample reaches a leaf, a descriptor of the sample is created by setting 1 to features that are active (value > 0), this descriptor is then compared to the leaf's descriptor (1 iff more than half of the training samples in the leaf are active in this feature). If more than 1/3 (empirically chosen) of the active features in the training descriptor are not active in the sample's descriptor, the sample is considered novel.

Test results: Given a single sample, this method shows poor novelty detection results. However, when collecting a group of samples that belong to the same unknown class, and using the median as a sample, we could achieve reasonable results, as shown in table 2. The classification results also improved drastically since the group traveled the tree in accordance with a majority vote.

TABLE 2
Novelty detection results – GMM-SVM-Tree

Test group size	Novel Error	Novel false positive	Novel false negative	Test Error (trained classes)
1	51.11	50	1.11	16.45
10	6.48	5	1.48	0.2

* The data is the 27 classes of the Caltech 256, the Novel classes are: goose, beer mug, vcr (11.11% of the classes).

4.2 Retrain

Given that the novel samples are known and for each novel class, the place in the tree to which it should be added is known (according to a tree trained with the novel class, e.g. goose as a brother of duck). We tested the retraining of the tree with only a few samples of the novel class. Our main goal was to test the knowledge transfer of the tree (e.g. if the tree learned how to distinguish birds, then this knowledge can be used for learning a new bird, using the tree hierarchy structure). We tested two methods: the first was splitting the destination leaf using an SVM from the novel samples; in the second method we also retrained every SVM classifier in the route from the root to the novel class' leaf.

TABLE 3
Novelty retrain Test Error – GMM-SVM-Tree

added group size	Tree leaf split	Tree full path fix	1-vs-Rest
0	25.56	25.56	23.14
1	25.37	25.74	23.33
5	23.52	23.70	20.37
10	21.67	21.48	18.70
All from start	17.22	17.22	15.18

* The data is the 27 classes of the Caltech 256, the Novel classes are: goose, beer mug, vcr (11.11% of the classes), the classifiers were trained on 24 classes, and then retrained with part of the 3 new classes.
TSVM/C-KMEANS trees produced comparable results.

In table 3 we compared the tree retrain methods to the retraining of a 1-vs-Rest SVM classifier with the added group size; it is evident that 1-vs-Rest SVM does not only produce a better classification rate, but has a higher benefit from the 5 added samples (~3% compared with the ~2% of the trees).

5. Summary

In this chapter we examined four hierarchical tree-like meta-structures, observing that large misclassification errors are performed for pairs of classes that are similar, both in the tree as well as in human perception. We sought to utilize these trees for information transfer to new classes of object. The first step was novelty detection; our method achieved good novelty detection rates, with a group of samples that belong to the same unknown class. However, yielded poor results with a single sample. The second step was using the tree to bootstrap the classification of the new classes. Our experiments reveal that 1-vs-Rest SVM produced favorable results compared to updating the trees. This means that the tree’s information transfer was unproductive.

References

Bo, L., Ren, X., and Fox, D. Multipath (2013). sparse coding using hierarchical matching pursuit. In CVPR.

Bodesheim, P., Freytag, A., Rodner, E., & Denzler, J. (2015). Local Novelty Detection in Multi-class Recognition Problems. In Applications of Computer Vision (WACV), 2015 IEEE.

Chapelle O., Sindhwani V., and Keerthi S.S. (2008). Optimization techniques for semi-supervised support vector machines. The Journal of Machine Learning Research, 9:203–233.

Coppi, D., de Campos, T., Yan, F., Kittler, J., & Cucchiara, R. (2014, April). On detection of novel categories and subcategories of images using incongruence. In Proceedings of International Conference on Multimedia Retrieval (p. 337). ACM.

Crammer, K., & Singer, Y. (2001). On the algorithmic implementation of multiclass kernel-based vector machines. Journal of Machine Learning Research, 2:265–292.

Dekel, O., Keshet, J., & Singer, Y. (2004, July). Large margin hierarchical classification. In Proceedings of the twenty-first international conference on Machine learning (p. 27). ACM.

Donahue J., Jia Y., Vinyals O., Hoffman J., Zhang N., Tzeng E., and Darrell T. (2014). DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. In ICML.

Everingham M., Van Gool L., Williams C. K. I., Winn J., and Zisserman A, (2012). The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., & Lin, C.-J. (2008). LIBLINEAR: A library for large linear classification. Journal of Machine Learning.

Fan, J., Zhang, J., Mei, K., Peng, J., & Gao, L. (2014). Cost-sensitive learning of hierarchical tree classifiers for large-scale image classification and novel category detection. Pattern Recognition.

Gammerman, A., Vovk, V., & Vapnik, V. (1998). Learning by transduction. In Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence (pp. 148-155). Morgan Kaufmann Publishers Inc.

Gao, T., and Koller D. (2011). Discriminative learning of relaxed hierarchy for large-scale visual recognition. In ICCV.

Griffin, G., Holub, A., and Perona, P. (2006). The caltech 256. In Caltech Technical Report.

Hu Y., Wang J., Yu N., Hua and X. (2008). Maximum margin clustering with pairwise constraints. In ICDM.

Hsu, C.-W., & Lin, C.-J. (2002). A Comparison of Methods for Multiclass. IEEE Transactions on Neural Networks, 13(2):415–425.

Hsu C.-W., Chang C.-C., and Lin C.-J. (2003). A practical guide to support vector classification. Technical report, Department of Computer Science, National Taiwan University.

Joachims Thorsten, (1999). Transductive Inference for Text Classification using Support Vector Machines. International Conference on Machine Learning (ICML).

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet clas- sification with deep convolutional neural networks. In NIPS.

Liu, D., Yan, S., Mu, Y., Hua, X.-S., Chang, S.-F., & Zhang, H.-J. (2011). Towards Optimal Discriminating Order for Multiclass Classification. In Proceedings of the IEEE International Conference on Data Mining.

Platt John C., Cristianini Nello, Shawe-Taylor John (2000), Large Margin DAGs for Multiclass Classification, MIT Press.

Razavian A., Azizpour H., Sullivan J., and Carlsson S., (2014). CNN Features off-the-shelf: an Astounding Baseline for Recognition, CoRR, vol. bs/1403.6382.

Rohrbach, Marcus, Stark Michael, Schiele Bernt (2011), Evaluating Knowledge Transfer and Zero-Shot Learning in a Large-Scale Setting, In CVPR.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., & LeCun, Y. (2014). Overfeat: Integrated recognition, localization and detection using convolutional networks. ICLR.

Shental, N., Hertz, T., Bar-Hilel, A., & Weinshall, D. (2003). Computing ganssian mixture models with EM using equivalence constraints.

Wager S., Wang S., and Liang P. (2013). Dropout training as adaptive regularization. In Advances in Neural Information Processing Systems 26, pages 351–359.

Wagstaff K., Cardie C. (2001). Rogers S., and Schroedl S. Constrained k-means clustering with background knowledge. In Proc. 18th International Conference on Machine Learning.

Weinshall, D., Hermansky, H., Zweig, A., Luo, J., Jimison, H., Ohl, F., & Pavel, M. (2008). Beyond novelty detection: Incongruent events, when general and specific classifiers disagree. In Advances in Neural Information Processing Systems (pp. 1745-1752).

Weston, J., & Watkins, C. (1998). Multi-class support vector machines. Technical Report CSD-TR-98-04, Department of Computer Science, Royal Holloway, University of London.

Yang, X., Lv, F., Cai, L., & Li, D. (2014). Adaptive learning region importance for region-based image retrieval. IET Computer Vision.

Zeiler M. D. and Fergus R. (2013). Visualizing and understanding convolutional networks. CoRR, abs/1311.2901.