Three Guidelines of Online Learning for Large-Scale Visual Recognition

Yoshitaka Ushiku, Masatoshi Hidaka, Tatsuya Harada
The University of Tokyo
7-3-1 Hongo Bunkyo-ku, Tokyo Japan
{ushiku, hidaka, harada}@mi.t.u-tokyo.ac.jp

Abstract

In this paper, we would like to evaluate online learning algorithms for large-scale visual recognition using state-of-the-art features which are preselected and held fixed. Today, combinations of high-dimensional features and linear classifiers are widely used for large-scale visual recognition. Numerous so-called mid-level features have been developed and mutually compared on an experimental basis. Although various learning methods for linear classification have also been proposed in the machine learning and natural language processing literature, they have rarely been evaluated for visual recognition.

Therefore, we give guidelines via investigations of state-of-the-art online learning methods of linear classifiers. Many methods have been evaluated using toy data and natural language processing problems such as document classification. Consequently, we gave those methods a unified interpretation from the viewpoint of visual recognition. Results of controlled comparisons indicate three guidelines that might change the pipeline for visual recognition.

1. Introduction

By virtue of recent advances in computer science and because of the culture of sharing of multimedia information such as photographs, vast quantities of labeled images have been used for visual recognition [14, 26, 34, 39]. Combinations of high-dimensional features and linear classifiers have been especially studied. Such high-dimensional features are generated from each image by pooling many mid-level features. Each mid-level feature is coded from a local descriptor. Recently, many techniques for coding and pooling have been proposed and compared using well-known datasets [3, 6].

During the last decade, numerous online learning methods for linear classification have also been widely studied [1, 8, 10, 11, 12, 33, 39] to address vast quantities of data which cannot be loaded on RAM at a time. Given the t-th training sample, $\boldsymbol{x}_t \in \mathbb{R}^d$, associated with a label, $y_t \in \mathcal{Y} = \{y_1, \ldots, y_m\}$, the sample is classified with the present weight vector, $\boldsymbol{\mu}_t^{y_i}$,[1] as $\hat{y_t} = \mathrm{argmax}_y \boldsymbol{\mu}_t^y \cdot \boldsymbol{x}_t$. The classifiers suffer from a loss when they misclassify the datum and get updated as $\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t + \tau_t \boldsymbol{x}_t$, where τ_t determines the step size. Because learning can be performed by holding one datum, online learning methods are appropriate for large-scale problems. Additionally, state-of-the-art methods outperform batch learning methods such as Support Vector Machine (SVM) as reported in [9, 16].

1. Perceptron can compete against the latest methods.
   - Provided that the second guideline is observed.
2. Averaging is necessary for any algorithm.
   - First-order algorithms w/o averaging cannot compete against second-order algorithms.
   - When averaging is used, the accuracies of all algorithms become very close to each other.
   - Averaging accelerates not only first-order algorithms but also second-order algorithms.
3. Investigate multiclass learning first.
   - Both one-versus-the-rest learning and multiclass learning achieve similar accuracy.
   - However, one-versus-the-rest takes much longer CPU time to converge than multiclass does.

Figure 1. Three guidelines for online learning for large-scale visual recognition.

Despite progress in online learning methods, few evaluations of these methods have been reported for large-scale visual recognition. Almost all approaches to obtain linear classifiers have been online versions of SVM [1, 2, 36] with a one-versus-the-rest (OVR) manner. Furthermore, the original SVM is not a multiclass classifier. With OVR, we divide training samples into a positive class or a negative class for each label. Then we train binary SVM for each

[1] Here, bias b is included in $\boldsymbol{\mu}_t$ as $\boldsymbol{\mu}_t^\top \leftarrow [\boldsymbol{\mu}_t^\top, b]$ by redefining $\boldsymbol{x}_t^\top \leftarrow [\boldsymbol{x}_t^\top, 1]$.

label. When we use OVR, however, the quantities of samples in the two classes (positive and negative) are imbalanced. Moreover, learning with OVR takes more CPU time because OVR might update many more weights than multiclass (MUL) in each step.

Our intuition tells us that the newest learning method performs best for large-scale visual recognition. However, this intuition is doubtful because the online learning methods have not been evaluated nor compared for visual recognition. Most online learning methods are evaluated using synthetic datasets and natural language datasets such as document classification.

As described in this paper, to give guidelines to choose learning methods for large-scale visual recognition, we investigate state-of-the-art online learning methods over various mid-level features using large-scale datasets.

Our main contributions are summarized as follows:

- To evaluate and discuss online learning algorithms on unified combinations of mid-level features and local descriptors.
- To propose averaged and sample-reweighted versions of second-order algorithms [10, 11, 12].
- To provide a source code, including all online learning algorithms compared in this paper.[2]

The remainder of this paper is organized as follows: Sec. 2 introduces related works for large-scale visual recognition. In Sec. 3, we overview the state-of-the-art algorithms from various perspectives. Qualitative and quantitative discussions are given, respectively, in Sec. 4 and Sec. 5. From these discussions, the three guidelines in Fig. 1 are obtained. Finally, we conclude this paper in Sec. 6.

2. Related Works

To achieve generic object recognition, large datasets are required because numerous objects with various appearances must be assessed. For example, with ImageNet [15], there are 14 million images for the 20,000 words in WordNet. Therefore, scalability of the data amount is necessary.

For scalability, combinations of high-dimensional features and linear classifiers have been widely studied [26, 34]. Mid-level features have been improved from traditional Bag-of-Visual-Words (BoVW) models [13]. Although a BoVW vector for each descriptor has only one non-zero element, recent mid-level features extract richer information: second-moment [31], first-moment [22, 42], and zero-moment [37, 40, 41] with respect to between descriptors and code words. Those features have been compared with common datasets. Some studies [3, 6] have compared the conventional features in a unified evaluation setting.

[2] http://www.mi.t.u-tokyo.ac.jp/static/projects/mil_averaged_learning/

Perceptron [33] has started the development of online learning algorithms for linear classification. As described in Sec. 3, these algorithms are divisible into two groups: first-order methods and second-order methods. Perceptron and gradient-based online SVMs [1, 2, 18, 36] are first-order methods. Recently, second-order algorithms [5, 9, 10, 11, 12, 16, 17] have been studied thoroughly for adaptive updates for each dimension using second-order information. They outperform batch SVM.

Although many proposals of mid-level features and their evaluations [3, 6] exist, few works describe investigations of learning methods for linear classifiers. In [4], only first-order algorithms and the averaging technique are evaluated. In [23], linear SVMs (including OVR and multiclass) have been investigated for large-scale problems. In [7], a link between Perceptron and SGD-SVM is discussed, but no quantitative comparison is included.

Furthermore, evaluations of these algorithms for visual recognition are rare. In [26, 30], some versions of SGD-SVM are evaluated with ImageNet [15]. [30] also proposed a reweighting OVR. In papers proposing the algorithms, the use of synthetic data and commonly used ML/NLP datasets is typically described. In natural language datasets, feature vectors are based on the Bag-of-Words (BoW) model, in which each dimension in the feature vector for each datum represents the presence of a certain word in the datum. In such cases, feature vectors tend to be sparse. Consequently, many researchers devote attention to adaptive learning when the occurrence ratio of each dimension of feature vectors differs. In visual recognition problems, however, feature vectors tend to be much denser than those of NLP problems.

3. Online Learning Algorithms

In this section, we introduce state-of-the-art online learning methods [1, 8, 10, 11, 12, 33] for linear classification. All update rules of the methods are summarized in Table 1.

Fundamentally, each method has been proposed as learning for binary classification. Two commonly used techniques apply binary classifiers themselves to multiclass problems. One is the one-versus-the-rest (OVR) technique described in Sec. 1. The other is the one-versus-one (OVO) technique. With the OVO, we train $m(m-1)/2$ classifiers for all pairs of labels. We use the OVR because the OVO requires numerous classifiers for labels of many kinds.

The overview of binary learning is shown in Fig. 2. In an online learning scheme, t-th sample, $\boldsymbol{x}_t$, is classified with the t-th weights, $\boldsymbol{\mu}_t$, by checking the sign of the inner product, $\boldsymbol{\mu}_t \cdot \boldsymbol{x}_t$. Samples are permuted randomly for stable learning because arranging samples in label order makes convergence slow. We can verify the prediction by checking if the margin, $\gamma_t = y_t(\boldsymbol{\mu}_t \cdot \boldsymbol{x}_t)$, is greater than zero because of the ground truth, $y_t = \pm 1$. However, we seek to enlarge

Initialize $\boldsymbol{\mu}_0 = 0$ and $\Sigma_0 = I$
while classifiers are not converged **do**
  for $t = 1, 2, \ldots, N$ **do**
    Receive sample $\boldsymbol{x}_t \in \mathbb{R}^d$.
    Predict $\hat{y}_t = \mathrm{sign}(\boldsymbol{\mu}_t \cdot \boldsymbol{x}_t)$.
    Get true label y_t and margin $\gamma_t = y_t(\boldsymbol{\mu}_t \cdot \boldsymbol{x}_t)$.
    if $\gamma_t < E$
      Set $\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t + \alpha_t y_t \Sigma_t \boldsymbol{x}_t$.
      Set $\Sigma_{t+1}^{-1} = \Sigma_t^{-1} + \beta_t \mathrm{diag}(\boldsymbol{x}_t)^2$.
    end if
  end for
end while

Figure 2. Overview of learning binary classifiers.

the margin γ_t for stable classification. Therefore, we check if $\gamma_t > E$, where $E \geq 0$.

We update $\boldsymbol{\mu}$ using a step size α_t as $\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t + \alpha_t y_t \boldsymbol{x}_t$ in first-order algorithms. Recent second-order algorithms use $\Sigma_t \in \mathbb{R}^{d \times d}$ as confidence information for the dimensions where non-zero values frequently appear not to be updated widely. We must learn $d \times d$ elements for each label if Σ_t is a full matrix. Therefore, diagonal matrices are commonly used. In Fig. 2 and Fig. 3, $\mathrm{diag}(\boldsymbol{x}_t)$ is a diagonal matrix having elements of $\boldsymbol{x}_t$ as diagonal elements.

An overview of multiclass (MUL) learning is portrayed in Fig. 3. Given the sample, $\boldsymbol{x}_t$, and its label, y_t, which now represents the label number, we treat a violating label, $y'_t = \arg\max_{y \in \mathcal{Y} \setminus y_t} \boldsymbol{\mu}_t^y \cdot \boldsymbol{x}_t$. Then we redefine the margin, $\gamma_t = \boldsymbol{\mu}_t^{y_t} \cdot \boldsymbol{x}_t - \boldsymbol{\mu}_t^{y'_t} \cdot \boldsymbol{x}_t$, and check if $\gamma_t > E$. Because almost all algorithms are proposed for binary classification, we modify them for multiclass learning in accordance with [8]. The details are described in Supplemental Materials.

Whether a classifier is for binary or for multiclass, the label for a sample $\boldsymbol{x}_t$ is predicted as $\hat{y}_t = \arg\max_y \boldsymbol{\mu}_t^y \cdot \boldsymbol{x}_t$.

3.1. Perceptron

Perceptron, which was proposed in [33] more than half a century ago, is a traditional algorithm to obtain a linear classifier. The last layer of Deep Convolutional Neural Network [24] is similar to the multiclass version of this algorithm. Margin γ_t is simply expected to be more than zero. Although the step size is also simply defined as $\alpha_t = 1$ in [33], we tune the fixed step size $\alpha_t = C$ for better accuracy.

3.2. Stochastic Gradient Descent SVM

The objective function of the original SVM, which is batch learning, is the following:

$$\boldsymbol{\mu} = \arg\min_{\boldsymbol{\mu}} \frac{1}{2}\|\boldsymbol{\mu}\|^2 + \sum_{n=1}^{N} \alpha_n \max\{0, 1 - \gamma_n\}, \qquad (1)$$

Initialize $\boldsymbol{\mu}_0 = 0$ and $\Sigma_0 = I$
while classifiers are not converged **do**
  for $t = 1, 2, \ldots, N$ **do**
    Receive sample $\boldsymbol{x}_t \in \mathbb{R}^d$.
    Predict $\hat{y}_t = \arg\max_{y \in Y}(\boldsymbol{\mu}_t^y \cdot \boldsymbol{x}_t)$.
    Get ...
      true label y_t,
      violating label $y'_t = \arg\max_{y \neq y_t}(\boldsymbol{\mu}_t^y \cdot \boldsymbol{x}_t)$,
      and their margin $\gamma_t = \boldsymbol{\mu}_t^{y_t} \cdot \boldsymbol{x}_t - \boldsymbol{\mu}_t^{y'_t} \cdot \boldsymbol{x}_t$.
    if $\gamma_t < E$
      Set $\boldsymbol{\mu}_{t+1}^{y_t} = \boldsymbol{\mu}_t^{y_t} + \alpha_t \Sigma_t^{y_t} \boldsymbol{x}_t$.
      Set $\boldsymbol{\mu}_{t+1}^{y'_t} = \boldsymbol{\mu}_t^{y'_t} - \alpha_t \Sigma_t^{y'_t} \boldsymbol{x}_t$.
      Set $(\Sigma_{t+1}^{y_t})^{-1} = (\Sigma_t^{y_t})^{-1} + \beta_t \mathrm{diag}(\boldsymbol{x}_t)^2$.
      Set $(\Sigma_{t+1}^{y'_t})^{-1} = (\Sigma_t^{y'_t})^{-1} + \beta_t \mathrm{diag}(\boldsymbol{x}_t)^2$.
    end if
  end for
end while

Figure 3. Overview of learning multiclass classifiers.

where N is the number of all training samples. This batch version of SVM can be converted to online learning methods by introducing stochastic gradient descent (SGD) [1, 2] or sub-gradient descent [36]. Pegasos [36] used a sub-gradient of the object function with a subset of training samples. When the size of this subset becomes one, the algorithm closely simulates the stochastic gradient descent (SGD-SVM) [1, 2] as:

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t - \alpha_t \nabla(\lambda\|\boldsymbol{\mu}\|^2 + \max\{0, 1 - \gamma_t\}), \qquad (2)$$

where λ is a hyperparameter that must be tuned manually. As described in [1], there is also a second-order version of SGD using an approximation of the Hessian for the objective function. In this paper, first-order SGD is used because the first-order version is commonly used for visual recognition [26, 30].

The most important problem is to design step size α_t. A usual technique [1, 2] states that $\alpha_t = 1/\lambda(t + t_0)$ with another hyperparameter t_0. The other [7, 30] is to define that $\alpha_t = C$ where $1 \gg C > 0$.

Another problem is related to regularization. We should tune the parameter λ. One empirical definition is $\lambda = 1/N$ where N is the data amount. Using a naive implementation, we must regularize all classifiers whether $1 - \gamma_t > 0$ or not. One famous approach is divide $\boldsymbol{\mu}_t$ into $l_t \boldsymbol{r}_t$, where l_t is the L_2 norm of $\boldsymbol{\mu}_t$ and $\boldsymbol{r}_t$ is normalized vector of $\boldsymbol{\mu}_t$. Consequently, regularization can be performed by multiplying l_t by $(1 - \alpha_t \lambda)$. However, [7] obviates regularization by defining $\lambda = 0$. Instead, the authors use early stopping, which is stopping training using a validation dataset. This version of SGD-SVM is the same as a variation of Perceptron called Margin Perceptron [19]. Indeed, [30] uses fixed step size

$\alpha_t = C$ and discards regularization for large-scale datasets. Experimental results in [30] show that SGD-SVM without regularization (Margin Perceptron) achieves similar or superior performance to SGD-SVM with L_2 regularization.

3.3. Passive–Aggressive

The largest benefit of Passive–Aggressive (PA) [8] is that the update coefficient is calculated analytically according to the loss. Here, we sought to decrease the hinge loss, $1 - \gamma_t$ and not to change the weight radically:

$$\boldsymbol{\mu}_{t+1} = \arg\min_{\boldsymbol{\mu}} \frac{1}{2}\|\boldsymbol{\mu} - \boldsymbol{\mu}_t\|^2 \text{ s.t. } 1 - \gamma_t = 0. \tag{3}$$

The equation presented above is the objective function of PA. *Objective functions of all other algorithms explained later are also defined as a form of each update.*

This is solvable analytically with ease. An important shortcoming is that $\boldsymbol{\mu}_{t+1}$ always classifies $\boldsymbol{x}_t$ as y_t, whether y_t is correct or not. It is impossible to design large datasets that include no label noise. In [8], therefore, aggressiveness parameter C is introduced to soften the condition:

$$\boldsymbol{\mu}_{t+1} = \arg\min_{\boldsymbol{\mu}} \frac{1}{2}\|\boldsymbol{\mu} - \boldsymbol{\mu}_t\|^2 + C(1 - \gamma_t). \tag{4}$$

This version is called PA-I [8]. Another version, PA-II, uses the squared hinge loss, $C(1 - \gamma_t)^2$, in the second term. According to [8], the accuracies of PA-I and PA-II are mutually close. For these analyses, we used PA-I as PA.

3.4. Confidence-Weighted

The main difference between Confidence-Weighted (CW) [16] and PA is that CW has the confidence weight Σ, a diagonal $d \times d$ matrix. If a classifier learns about a certain dimension of feature vectors many times, then the classifier must be more confident about that dimension. In other words, the classifier is expected to update less confident dimensions larger. Such an adaptive update makes convergence faster than the first-order algorithms.

Therefore, CW considers weights as a normal distribution, $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$. We expect that the $(t+1)$-th weight from $\mathcal{N}(\boldsymbol{\mu}_{t+1}, \Sigma_{t+1})$ can classify $\boldsymbol{x}_t$ correctly with a fixed probability, η. This condition is expressed as $\gamma_t \geq \phi\sqrt{v_t}$, where $v_t = \boldsymbol{x}_t^\top \Sigma_t \boldsymbol{x}_t$, and $\phi = \Phi^{-1}(\eta)$. Φ is the cumulative function of the normal distribution.

To preserve the current classifiers, Kullback–Leibler divergence is used instead of the squared L_2 norm, $\|\boldsymbol{\mu} - \boldsymbol{\mu}_t\|^2$, in PA. Consequently, the objective function is the following:

$$\begin{aligned}(\boldsymbol{\mu}_{t+1}, \Sigma_{t+1}) &= \arg\min_{\boldsymbol{\mu},\Sigma} \mathrm{D}_{\mathrm{KL}}\left(\mathcal{N}(\boldsymbol{\mu}, \Sigma) \| \mathcal{N}(\boldsymbol{\mu}_t, \Sigma_t)\right) \\ &\text{s.t.} \quad \phi\sqrt{v_t} - \gamma_t = 0.\end{aligned} \tag{5}$$

In [16], the solution was approximated whereas exact updates for binary and multiclass classifications were proposed, respectively, in [10] and [9].

3.5. Adaptive Regularization of Weight

The salient shortcoming of CW is its poor adaptability to label noise. Therefore, Adaptive Regularization of Weight (AROW) [11] introduces the squared hinge loss as:

$$\begin{aligned}(\boldsymbol{\mu}_{t+1}, \Sigma_{t+1}) = \arg\min_{\boldsymbol{\mu},\Sigma} &\mathrm{D}_{\mathrm{KL}}\left(\mathcal{N}(\boldsymbol{\mu}, \Sigma) \| \mathcal{N}(\boldsymbol{\mu}_t, \Sigma_t)\right) \\ &+ C(1 - \gamma_t)^2 + C\boldsymbol{x}_t^\top \Sigma \boldsymbol{x}_t,\end{aligned} \tag{6}$$

where C is a constant parameter to be tuned.[3] The third term aims to converge classifiers faster.

3.6. Gaussian Herding

Gaussian Herding (NHERD) [12] is a modified version of PA for second-order algorithms. As is true also for CW and AROW, weights in HERD are expressed with normal distributions, $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$. Additionally, the t-th update is defined as a linear transformation of the distributions with matrices A_t. As a result, we obtained the objective function as:

$$\begin{aligned}(\boldsymbol{\mu}_{t+1}, A_t) = \arg\min_{\boldsymbol{\mu},A} &\frac{1}{2}(\boldsymbol{\mu} - \boldsymbol{\mu}_t)^\top \Sigma_t^{-1}(\boldsymbol{\mu} - \boldsymbol{\mu}_t) \\ &+ \frac{1}{2}\mathrm{Tr}\left((A - I)^\top \Sigma_t^{-1}(A - I)\Sigma_t\right) \\ &+ C(1 - \gamma_t)^2 + \frac{C}{2}\boldsymbol{x}_t^\top A \Sigma_t A^\top \boldsymbol{x}_t.\end{aligned} \tag{7}$$

3.7. Soft Confidence-Weighted

Soft Confidence-Weighted (SCW) [38] solves the problem of CWs poor adaptability to label noise by softening the condition according to the manipulation in PA [8]:

$$\begin{aligned}(\boldsymbol{\mu}_{t+1}, \Sigma_{t+1}) = \arg\min_{\boldsymbol{\mu},\Sigma} &D_{KL}\left(\mathcal{N}(\boldsymbol{\mu}, \Sigma) \| \mathcal{N}(\boldsymbol{\mu}_t, \Sigma_t)\right) \\ &+ C\left(\phi\sqrt{v_t} - \gamma_t\right).\end{aligned} \tag{8}$$

The difference between Eq. (5) and Eq. (8) is the same as the difference between Eq. (3) and Eq. (4); the condition is added to the equation with aggressiveness parameter C. Therefore, there are two versions of SCW: SCW-I and SCW-II. In particular, the last term of Eq. (8) would be $C(\phi\sqrt{v_t} - \gamma_t)^2$ for SCW-II. For this study, we used SCW-I as SCW.

4. Common Qualitative Issues

Update rules of all learning methods are presented in Table 1. Again, it is noteworthy that objective functions of PA, CW, AROW, NHERD, and SCW are already shown in their subsections. These algorithms are designed using a form of each update while batch SVM is designed using a total loss. In this section, we investigate two common issues.

[3] In [11], C is expressed as $\frac{1}{2r}$, where r is also a parameter to be tuned. Here, we use C for unified description.

Table 1. Update rules. Variables in Fig. 2 and Fig. 3 are shown in the middle columns. The last column shows the parameters to be tuned. For SCW, we show a slightly different form of update from that described in [38]. Details are described in Supplemental Materials. For binary classification, $\gamma_t = y_t(\boldsymbol{\mu}_t \cdot \boldsymbol{x}_t)$, $v_t = \boldsymbol{x}_t^\top \Sigma_t \boldsymbol{x}_t$, and $l(\boldsymbol{x}_t)^2 = \|\boldsymbol{x}_t\|^2$. For multiclass classification, $\gamma_t = \boldsymbol{\mu}_t^{y_t} \cdot \boldsymbol{x}_t - \boldsymbol{\mu}_t^{y_t'} \cdot \boldsymbol{x}_t$, $v_t = \boldsymbol{x}_t^\top (\Sigma_t^{y_t} + \Sigma_t^{y_t'}) \boldsymbol{x}_t$, and $l(\boldsymbol{x}_t)^2 = 2\|\boldsymbol{x}_t\|^2$. Whether the classification is binary or multiclass, $\phi = \Phi^{-1}(\eta)$ (Φ is the cumulative function of the normal distribution), $\psi = 1 + \phi^2/2$, and $\zeta = 1 + \phi^2$.

Method	E	α_t	β_t	Parameters
Perceptron [33]	0	C	0	C
SGD-SVM [1]	1	C	0	C
PA [8]	1	$\min\left\{C, (1-\gamma_t)/l(\boldsymbol{x}_t)^2\right\}$	0	C
CW [10]	$\phi\sqrt{v_t}$	$\max\left\{0, \frac{1}{v_t\zeta}\left(-\gamma_t\psi + \sqrt{\gamma_t^2\frac{\phi^4}{4} + v_t\phi^2\zeta}\right)\right\}$	$\frac{2}{-v_t+\sqrt{v_t^2+4v_t/(\alpha_t^2\phi^2)}}$	η
AROW [11]	1	$(1-\gamma_t)/(v_t + 1/C)$	C	C
NHERD [12]	1	$(1-\gamma_t)/(v_t + 1/C)$	$2C + C^2 v_t$	C
SCW [38]	$\phi\sqrt{v_t}$	$\min\left\{C, \max\left\{0, \frac{1}{v_t\zeta}\left(-\gamma_t\psi + \sqrt{\gamma_t^2\frac{\phi^4}{4} + v_t\phi^2\zeta}\right)\right\}\right\}$	$\frac{2}{-v_t+\sqrt{v_t^2+4v_t/(\alpha_t^2\phi^2)}}$	C, η

4.1. OVR vs. MUL

Two common choices of OVR are e-OVR, for which the number of negative samples is the same as the number of positive samples, and u-OVR, for which all samples in other classes are selected as negative samples. In [30], reweighting samples for OVR (w-OVR) is proposed. In each iteration,[4] the number of negative samples is limited with respect to the number of positive samples.

[30] experimentally compared MUL and OVR using SGD-SVM. As a result, MUL outperformed e-OVR and u-OVR. Moreover, w-OVR outperformed multiclass SGD-SVM. Authentically, OVR is easily parallelized because a classifier for each label can be learned independently. However, MUL can also be parallelized easily [21]. In general, more accurate classifiers with less CPU time for learning are preferred. Herein, we compare these OVRs and MUL using state-of-the-art online learning methods.

4.2. Averaging

With most algorithms, training samples that are learned later strongly influence the classifiers. In [20], the weighted sum of $\boldsymbol{\mu}_{1\ldots T}$ is proposed for testing. Particularly, averaging them as $\bar{\boldsymbol{\mu}} = \frac{1}{T}(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2 + \cdots + \boldsymbol{\mu}_T)$ greatly facilitates convergence. Because averaging naively takes much time, we used an efficient calculation described in Supplemental Materials. The authors of [32] insist that averaging is a kind of approximation of second-order algorithm. Indeed, averaged (first-order) SGD is proved to be an approximation of Newton-like second-order SGD in [32].

In [26], averaging SGD-SVM outperforms SGD-SVM for visual recognition. However, averaging weights using other state-of-the-art online learning methods are rarely evaluated for visual recognition.

[4] Here iteration means learning through all T samples.

5. Experiments on ImageNet

This section shows highlights of experimentally obtained results. Full results are provided in Supplemental Materials. For general evaluation, we used various subsets of ImageNet [15]. Namely, the dataset of ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2010, the subset of ILSVRC 2010 dataset, and the subset of ILSVRC 2012 dataset[5]. Both ILSVRC 2010 and 2012 datasets include 1.2 million training images, 50,000 validation images, and 150,000 testing images for different sets of 1000 classes.

Classifiers trained with ILSVRC 2010 datasets are evaluated using 150,000 test samples. For both ILSVRC 2010 subsets and ILSVRC 2012 subsets, 100 training images were extracted from each class. ILSVRC 2012 has test samples, but the ground truth is not provided. Therefore, 5000 validation samples were used for validation. The rest were used for testing.

Parameters of online learning methods were tuned using validation data as follows: Cs in Perceptron, SGD-SVM, PA, CW, AROW, NHERD and SCW were determined by selecting the best one from $\{2^{-4}, 2^{-2}, 2^0, 2^2, 2^4\}$, ηs in CW and SCW was determined by selecting from $\{0.5, 0.6, \ldots, 0.9\}$. Fundamentally, we allowed at most 10 passes over the data set for the best accuracy. We repeated all evaluations five times. Herein, we present the results obtained using the mean and the standard deviation.

5.1. ILSVRC 2010 Dataset

For this dataset, SIFT [27] descriptor and Fisher Vector (FV) [31] were used. We extracted each descriptor from a regular grid with step size 6 pixels at multiple patch sizes: 16×16, 25×25, 36×36, 49×49, and 64×64. As a result, tens of thousands of descriptors were extracted for each im-

[5] http://www.image-net.org/download

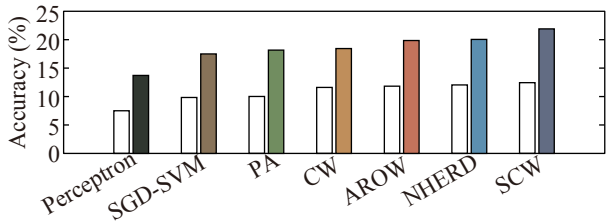

Figure 4. Comparison using ILSVRC 2010 1.2M dataset with SIFT+FV. White bars are performances using a 100k subset of ILSVRC 2010.

age. For FV, according to [30], we reduced the dimensions of SIFT to 64 using PCA, and obtained a Gaussian Mixture Model (GMM) with 16 components.

5.2. ILSVRC 2012 Dataset

We used four local descriptors: SIFT [27], Local Binary Pattern (LBP) [28], GIST [29], and CSIFT [35], and three mid-level features: BoVW, Locality-constrained Linear Coding (LLC) [37], and FV [31]. The size of patch and the grid width are the same as SIFT from ILSVRC 2010. Our main contribution is to compare online learning algorithms. Comparison with these mid-level features are done in [6], although we also slightly contribute by comparing them using various local descriptors.

LBP [28] is extracted from 2×2 cells in each patch. From each cell, a histogram of 256 bins of local patterns is extracted. Combinations of LBP and gradient based descriptor, such as SIFT, are shown to be effective for large-scale visual recognition [26]. GIST [29] is extracted from 4×4 cells in each patch. From each cell, responses from 20 Gabor filters are extracted on R, G and B channels. Usually, GIST is used for a global feature. This report is the first describing the use of GIST as a local descriptor for mid-level features. CSIFT, one variation of color SIFTs, is a 384-dimensional descriptor that has been shown to perform best with BoVW for visual recognition in [35].

Each mid-level feature is generated from each descriptor. For FV, we reduced the dimensions of descriptors to 64 using PCA, and obtained a GMM with 256 components. For BoVW and LLC, we learned 2048 codewords using k-means. BoVW and LLC were calculated respectively over 1×1, 2×2, and 3×1 cells according to Spatial Pyramid Matching [25]. Additionally, we reduced the memory usage with Product Quantization [22] according to [34]. We divide mid-level features into each of eight dimension vectors, and generate 256 clusters using k-means. All FVs of training samples are quantized and approximated with the centroids of each cluster when learning.

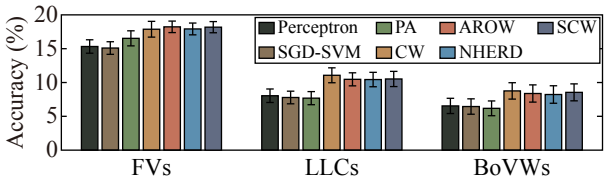

Figure 5. Comparison using ILSVRC 2012 subset. Each bar represents the mean accuracy among mid-level features from four descriptors. For example, the accuracy using FVs is the mean of the accuracies using SIFT+FV, LBP+FV, GIST+FV and CSIFT+FV.

5.3. Result 1: Accuracies of not Averaged Classifiers

Because of space constraints, we present summarized results here. For details, see the numerical results shown in the Supplemental Material.

To compare all algorithms explained in Sec. 3, we first used ILSVRC 2010 dataset. In Fig. 4 the accuracies of all algorithms without averaging are shown[6]. We found the same trend both in the 1.2M images and in the 100k images: The second-order algorithms (right four algorithms in Fig. 4) tend to outperform first-order algorithms.

To increase the reliability of our evaluations, we assessed all algorithms using the ILSVRC 2012 subset. Moreover, we investigated 12 combinations of local descriptors and mid-level features including SIFT+FV. Figure 5 represents the accuracies of all algorithms for each mid-level feature. To compare the algorithms easily, we averaged four accuracies from the same mid-level features generated from four descriptors. Again, we can find the superiority of the second-order algorithms. Particularly CW performs best on nine combinations among all 12 combinations of descriptors and mid-level features. AROW and SCW perform best on the three remaining combinations.

5.4. Result 2: Averaging Does Boost All

When we compared all algorithms without averaging, the second-order algorithms simply seemed to outperform the first-order algorithms. However, Fig. 6 shows that averaging dramatically eliminates the difference of accuracies. The accuracies of second-order algorithms are also boosted.

For a comparison of several datasets, we again evaluated the algorithms with averaging on ILSVRC 2012 subset. Figure 7 shows the accuracies of all algorithms with averaging for each mid-level feature. Results show three facts that are little-noted in the literature, although averaging itself is a well-known technique. First, second-order algorithms are also boosted for all combinations. Secondly, when averaging is used, SCW turns to perform best instead of CW with

[6] In [30], the accuracy of SIFT+FV with the same parameter is around 25%. The accuracies in Fig. 4 are slightly different, probably because of the difference of SIFT extraction and GMM training.

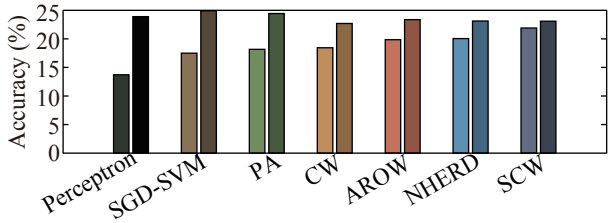

Figure 6. Comparison using ILSVRC 2010 1.2M dataset with SIFT+FV. The darker bar for each algorithm shows the accuracy with averaging. The brighter shows the accuracy without averaging for easy reference.

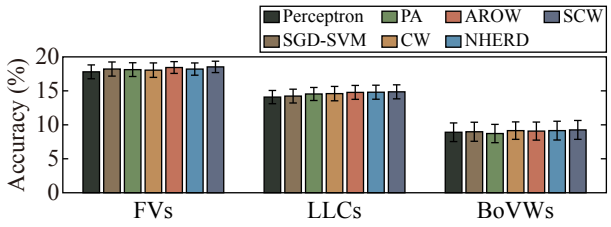

Figure 7. Comparison using ILSVRC 2012 subset. Each bar represents the mean accuracy among mid-level features from four descriptors.

many combinations of mid-level features and descriptors. Thirdly, however, the differences among all algorithms are narrowed again. As a result, first-order algorithms such as Perceptron, SGD-SVM and PA achieve comparative performance using several combinations including the result obtained using ILSVRC 2010 dataset. Here, we conclude the first guideline: Perceptron can compete against the latest algorithms, only when averaging is employed.

The next question is whether the averaging just hasten the convergence. Because online learning algorithms are evaluated in ten iterations, we also stop learning earlier than in the tenth iteration in almost all experiments. To do justice to this inquiry, we continue learning until the 50th iteration on the ILSVRC 2010 dataset.

Figure 8 shows the convergence of Perceptron and SCW. Note that the accuracies are evaluated not using training data but using test data. After some iterations, both averaged and not-averaged classifiers seem to reduce their performance on test data. This is true mainly because of an overfit to the training data. Furthermore, the best accuracy of the averaged classifier is better than the not-averaged classifiers, even with many iterations. Averaging not only accelerates the optimization but also improves the generalization accuracy. Therefore, we propose the second guideline: averaging is necessary for any algorithm.

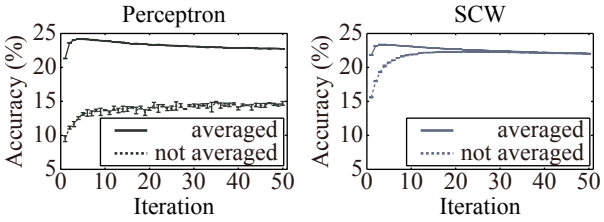

Figure 8. Comparison about averaging. For other algorithms, see Supplemental Material.

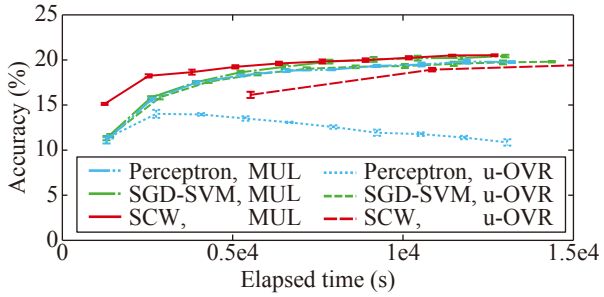

Figure 9. Comparison of MUL and OVRs. For other algorithms, see Supplemental Material.

5.5. Result 3: MUL vs. OVRs

In Fig. 9, the relation between elapsed time for learning and accuracy using MUL and all OVRs on SIFT+FV of ILSVRC 2012 is shown. First, Perceptron shortly begins to overfit with OVRs. Secondly, especially for second-order algorithms, MUL can converge faster than OVR. This is true mainly because updating the weights becomes the rate-limiting step. Prediction with current weights is rate-limiting for first-order algorithms. Therefore, we propose the third guideline: investigate multiclass learning first.

6. Conclusions

To realize generic object recognition, a large amount of data is required. Considering scalability, combinations of mid-level features and online learning for linear classifiers are suitable for large-scale visual recognition.

As described in this paper, we gave qualitative and quantitative comparisons of these online learning algorithms. To date, no report has described a study investigating state-of-the-art algorithms for visual recognition or a study evaluating those algorithms in unified experimental settings. When these algorithms were proposed, toy data and the NLP dataset were used for evaluation. Comparison using conventional settings for visual recognition must be conducted. This paper presents three guidelines based on results of image classification: 1) Perceptron can compete against the latest algorithms; 2) Averaging is necessary for any algorithm; 3) Investigate multiclass learning first.

References

[1] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *COMPSTAT*, 2010. 1, 2, 3, 5

[2] L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In *NIPS*, 2007. 1, 2, 3

[3] Y.-L. Boureau, F. Bach, Y. LeCun, and J. Ponce. Learning mid-level features for recognition. In *CVPR*, 2010. 1, 2

[4] V. R. Carvalho and W. W. Cohen. Single-pass online learning: Performance, voting schemes and online feature selection. In *KDD*, 2006. 2

[5] N. Cesa-Bianchi, A. Conconi, and C. Gentile. A second-order perceptron algorithm. *SICOMP*, 34(3):640–668, 2005. 2

[6] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. In *BMVC*, 2011. 1, 2, 6

[7] R. Collobert and S. Bengio. Links between perceptrons, mlps and svms. In *ICML*, 2004. 2, 3

[8] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *JMLR*, 7:551–585, 2006. 1, 2, 3, 4, 5, 9

[9] K. Crammer, M. Dredze, and A. Kulesza. Multi-class confidence weighted algorithms. In *EMNLP*, 2009. 1, 2, 4

[10] K. Crammer, M. Dredze, and F. Pereira. Exact convex confidence-weighted learning. In *NIPS*, 2008. 1, 2, 4, 5

[11] K. Crammer, A. Kulesza, and M. Dredze. Adaptive regularization of weight vectors. In *NIPS*, 2009. 1, 2, 4, 5

[12] K. Crammer and D. D. Lee. Learning via gaussian herding. In *NIPS*, 2010. 1, 2, 4, 5, 9

[13] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *ECCV Workshop on SLCV*, 2004. 2

[14] J. Deng, A. C. Berg, K. Li, and L. Fei-Fei. What does classifying more than 10,000 image categories tell us? In *ECCV*, 2010. 1

[15] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 2, 5

[16] M. Dredze, K. Crammer, and F. Pereira. Confidence-weighted linear classification. In *ICML*, 2008. 1, 2, 4

[17] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. In *COLT*, 2010. 2

[18] J. Duchi and Y. Singer. Efficient online and batch learning using forward backward splitting. *JMLR*, 10:2899–2934, 2009. 2

[19] R. O. Duda and P. E. Hart. *Pattern classification and scene analysis*. Wiley, 1973. 3

[20] Y. Freund and R. E. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296, 1999. 5

[21] K. B. Hall, S. Gilpin, and G. Mann. Mapreduce/bigtable for distributed optimization. In *NIPS Workshop LCCC*, 2010. 5

[22] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *CVPR*, 2010. 2, 6

[23] S. S. Keerthi, S. Sundararajan, K.-W. Chang, C.-J. Hsieh, and C.-J. Lin. A sequential dual method for large scale multi-class linear svms. In *KDD*, 2008. 2

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 3

[25] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006. 6

[26] Y. Lin, F. Lv, S. Zhu, M. Yang, T. Cour, K. Yu, L. Cao, and T. Huang. Large-scale image classification: Fast feature extraction and svm training. In *CVPR*, 2011. 1, 2, 3, 5, 6

[27] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004. 5, 6

[28] T. Ojala, M. Pietikäinen, and D. Harwood. A comparative study of texture measures with classification based on feature distributions. *Pattern Recognition*, 29(l):51–59, 1996. 6

[29] A. Oliva and A. Torralba. Modeling the shape of the scene : A holistic representation of the spatial envelope. *IJCV*, 42(3):145–175, 2001. 6

[30] F. Perronnin, Z. Akata, Z. Harchaoui, and C. Shmid. Towards good practice in large-scale learning for image classification. In *CVPR*, 2012. 2, 3, 4, 5, 6

[31] F. Perronnin, J. Sánchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. In *ECCV*, 2010. 2, 5, 6

[32] B. T. Polyak and A. B. Juditsky. Acceleration of stochastic approximation by averaging. *SICON*, 30(4):838–855, 1992. 5

[33] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization. *Psychological Review*, 65(6):386–408, 1958. 1, 2, 3, 5

[34] J. Sánchez and F. Perronnin. High-dimensional signature compression for large-scale image classification. In *CVPR*, 2011. 1, 2, 6

[35] K. E. A. v. d. Sande, T. Gevers, and C. G. M. Snoek. Evaluating color descriptors for object and scene recognition. *TPAMI*, 32(9):1582–1596, 2010. 6

[36] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for svm. In *ICML*, 2007. 1, 2, 3

[37] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *CVPR*, 2010. 2, 6

[38] J. Wang, P. Zhao, and C. S. Hoi. Exact soft confidence-weighted learning. In *ICML*, 2012. 4, 5, 9

[39] J. Weston, S. Bengio, and N. Usunier. Wsabie: Scaling up to large vocabulary image annotation. In *IJCAI*, 2011. 1

[40] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, 2009. 2

[41] K. Yu, T. Zhang, and Y. Gong. Nonlinear learning using local coordinate coding. In *NIPS*, 2009. 2

[42] X. Zhou, K. Yu, T. Zhang, and T. S. Huang. Image classification using super-vector coding of local image descriptors. In *ECCV*, 2010. 2