Fast training of object detection using stochastic gradient descent
Wijnhoven, R.G.J.; de With, P.H.N.

Published in:
Proceedings of the 20th International Conference on Pattern Recognition (ICPR), 23-26 August 2010, Istanbul, Turkey

DOI:
10.1109/ICPR.2010.112

Published: 01/01/2010

Please check the document version of this publication:

- A submitted manuscript is the author's version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.
Fast Training of Object Detection using Stochastic Gradient Descent

Rob G.J. Wijnhoven\textsuperscript{1,2} \hspace{1cm} Peter H.N. de With\textsuperscript{2,3}
\textsuperscript{1} ViNotion BV \hspace{1cm} \textsuperscript{2} Univ. of Technol. Eindhoven / \textsuperscript{3} CycloMedia
\textit{Eindhoven, The Netherlands}

Abstract

Training datasets for object detection problems are typically very large and Support Vector Machine (SVM) implementations are computationally complex. As opposed to these complex techniques, we use Stochastic Gradient Descent (SGD) algorithms that use only a single new training sample in each iteration and process samples in a stream-like fashion. We have incorporated SGD optimization in an object detection framework. The object detection problem is typically highly asymmetric, because of the limited variation in object appearance, compared to the background. Incorporating SGD speeds up the optimization process significantly, requiring only a single iteration over the training set to obtain results comparable to state-of-the-art SVM techniques. SGD optimization is linearly scalable in time and the obtained speedup in computation time is two to three orders of magnitude. We show that by considering only part of the total training set, SGD converges quickly to the overall optimum.

1. Introduction

Models for object detection models are based on describing appearance and spatial information of the objects. Bag-of-Words models only use appearance information [4] and implicit shape models [11] add spatial information. Sliding window classifiers [5] encode appearance information on a regular grid, thereby implicitly modeling spatial information. We consider the sliding window classifier as proposed by Dalal and Triggs [5] because of its simplicity and good performance (e.g. pedestrian detection [6]). In this approach, a window is shifted over the region and each position in the image is classified into object or background.

Since the trained classifier has to be applied to a very high number of image positions in the detection process, a linear classifier is preferred for its computational simplicity. To train such a linear classifier, a Support Vector Machine (SVM) [3] is often used, as it can handle high-dimensional feature vectors effectively. Although efficient training algorithms exist, such as based on the decomposition of the problem, the training of an SVM is a computationally complex optimization process (see Shalev-Shwartz et al. [12]). The training time is superlinear in the number of training samples, in practice, when using a constrained number of training samples, execution times range in the order of tens of seconds to several minutes.

Stochastic Gradient Descent (SGD) algorithms have been successfully used for the training of neural networks [9]. Bottou et al. [1] have expanded this concept for the purpose of training linear classifiers, using a regularization constraint (as in SVMs), and showed major speedups in computation time with no loss in classification performance on large-scale learning problems. Another SVM implementation is proposed by Shalev-Schwarz et al. [12]. In the context of object recognition, SGD has been used in multi-layer convolutional networks for training the features but not for training the classifier [10]. In this paper, we use a pre-defined feature transform and use SGD to train the final classifier, instead of the typically used Sequential Minimal Optimization (SMO). We show that it obtains similar classification performance compared to the state-of-the-art SVM implementations SVMLight [8] and libSVM [2], while gaining a speedup of two to three orders of magnitude in computation time. Furthermore, we evaluate the performance of SGD when only a part of the training set is presented to the training algorithm and show that it quickly converges.

2. Stochastic Gradient Descent

In a supervised learning problem, we are given a set of training samples \((x, y) \in \mathcal{X} \times \mathcal{Y}\), taken from the probability distribution \(P(x, y)\). The conditional probability \(P(y|x)\) represents the relationship between input vector \(x\) and output label \(y\) that we are trying to estimate. The difference between the estimated label \(\hat{y}\) and
the true label \( y \) is represented by a loss function \( l(\hat{y}, y) \). We try to estimate the function \( f \) that minimizes the expected risk

\[
E(f) = \int l(f(x), y)dP(x, y) = \mathbb{E}[l(f(x), y)].
\]

We are given \( n \) samples \((x_i, y_i), i = 1...n\) of the unknown distribution \( P(x, y) \). We try to find the function \( f_n \) that minimizes the empirical risk

\[
E_n(f_n) = \frac{1}{n} \sum_{i=1}^{n} l(f(x_i), y_i) = \mathbb{E}_n[l(f(x), y_i)].
\]

The function \( f \) is linearly parameterized by \( w \in \mathbb{R}^d \), \( d \) being the dimensionality of feature vectors \( x \). In standard GD techniques, the empirical risk is minimized by using the true gradient of the \( w \) vector, which is typically estimated as the sum of the gradients caused by each individual training sample.

\[
w_{t+1} = w_t - \frac{\eta}{n} \frac{\delta E_n}{\delta w}(w_t)
\]

\[
w_{t+1} = w_t - \eta \frac{1}{n} \sum_{i=1}^{n} \frac{\delta}{\delta w} l(f_t(x_i), y_i)
\]

Note that \( \eta \) is the update/gain factor or step size used to update the solution \( w_t \) at step \( t \). Standard GD requires one complete sweep over the training set in order to calculate the gradient and thus update the optimization parameters. Because many iterations might be required to reach the global optimum, this approach is impractical for large datasets. SGD considers one sample at each iteration and updates the weight vector \( w \) iteratively using a time-dependent weighting factor, leading to

\[
w_{t+1} = w_t - \eta \frac{\delta}{\delta w} l(f_t(x_t), y_t).
\]

Compared to GD, SGD requires much less time per update, resulting in faster convergence. Note that updates of the optimization parameter \( w_t \) are noisy because one sample is considered at a time.

In order to obtain a linear classifier that has a regularization constraint, as used in Support Vector Machines (SVM) [3], we use the Hinge loss SVM objective function \( l(z, y) = max\{0, 1 - yz\} \), with \( z = wx + b \) and minimize the function

\[
\frac{\lambda}{2} ||w||^2 + \frac{1}{n} \sum_{i=1}^{n} l(y_i(wx_i + b)),
\]

where the regularization term \( ||w|| \) represents the size of the margin in feature space. Embedding this function in the SGD framework results in

\[
w_{t+1} = w_t - \eta \left( \lambda w_t + \frac{\delta l(y_i(wx_i + b))}{\delta w_t}\right)
\]

with \( \eta_t = \frac{1}{\lambda(t + t_0)} \).

The parameters \( \lambda \) and \( t_0 \) have to be set by the user. Because \( \eta_0 = 1/(\lambda t_0) \), \( t_0 \) and \( \eta_0 \) are directly related.

3. Object Detection

In order to obtain an invariant object description, we use our implementation of the Histogram of Oriented Gradients (HOG) algorithm, as proposed by Dalal and Triggs [5]. We use the following parameters: cells of \( 8 \times 8 \) pixels, 4 block normalizations, 18 orientation bins using the sign, L2 feature normalization and a detector size of \( 104 \times 56 \) pixels. The dimensionality of the feature vector for each window becomes 6,552.

Object detection is obtained by sliding a window over the image and classifying the local description for each position into object/background. To detect objects of different size, the detection process is repeated for scaled versions of the input image. We use scale steps of 1.05. Finally, a mean-shift mode-finding algorithm merges window-level detections.

Dalal and Triggs [5] propose a two-stage approach to train a classifier for the object detection task. In the first stage, negative (background) samples are gathered by extracting a fixed number of features randomly from the set of background images. A first classifier is trained using all positive samples and the set of random negative samples. In the second stage, this initial classifier is used to classify every window position in the background images. All detections are added to the training set of negative samples. Because the classifier discards most of the background samples, only a limited set of new background samples is added. In a second stage, the final classifier is trained using both the positive samples, the random negative samples and the background samples from the first stage.

In our experiments, we focus on this second classifier training stage. We have trained an initial classifier by gathering 10 background samples randomly from each background image in the training set. Then, this classifier was applied to this background set and additionally found negative samples were added. The final training set contains 15k samples, of which only 500 are positive object samples.
4. Experiments and Results

We evaluate both the performance and execution times of the training algorithms for three implementations: \textit{libSVM} \cite{2}, \textit{SVMLight} \cite{8} and SGD. We employ the SGD implementation \textit{svmsgd2} by Bottou \cite{1} with some minor modifications. Bottou uses a heuristic to determine a good value for $t_0$ and the corresponding gain value $\eta_0$. We start with a gain value of $\eta_0 = 1$ by setting $t_0 = 1/\lambda$. In addition, we modify the gain factor for the updating of the bias term, that is also iteratively updated in the implementation. We set $\lambda$ to $1e^{-2}$ and $C$ for \textit{SVMLight} and \textit{libSVM} to $1.5e^{-2}$. We evaluate the performance of one epoch (a single sweep) over the training set. Because SGD expects random training samples, we shuffle the data randomly prior to training.

For our experiments, we use the PASCAL 2006 dataset \cite{7}, category \textit{Car}, because it is challenging and many results are provided in literature. The dataset includes 250 object images and by horizontal flipping 500 positive samples are obtained. There are 1,006 background images without objects. We use the validation set to tune our parameters (the window-level classification threshold) and test on the test set (2,686 images). Some detection examples are shown in Figure 1.

![Example detections on the PASCAL 2006 Car dataset.](image)

Since the optimization problem is similar for all implementations and they all reach a similar optimum, we omit the recall-precision curves because they nearly coincide and therefore directly list the detection scores in Table 1. Both the Area Under Curve (AUC) and the Average Precision (AP) measures as used by PASCAL are shown.

<table>
<thead>
<tr>
<th>Name</th>
<th>AUC</th>
<th>AP</th>
<th>sec</th>
<th>speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGD</td>
<td>45.6%</td>
<td>47.0%</td>
<td>0.15</td>
<td>$2.9 \times 10^4$</td>
</tr>
<tr>
<td>SVMLight</td>
<td>44.5%</td>
<td>43.1%</td>
<td>22</td>
<td>$1.5 \times 10^2$</td>
</tr>
<tr>
<td>libSVM</td>
<td>44.4%</td>
<td>43.1%</td>
<td>430</td>
<td>1</td>
</tr>
</tbody>
</table>

Note that the bias is also stochastically estimated by the SGD algorithm and does not always give the most satisfactory decision threshold for the window-level classifier, requiring some user interaction per detection problem to obtain optimal detection results. Note that for our execution of \textit{SVMLight}, we also adjust the threshold to achieve optimal performance (when using the bias from the optimization process over the manually set value, the AUC drops from 44.5\% to 34.7\%).

Figure 2 visualizes the update process of SGD optimization over time. The update of the weight vector $w$ is depicted in Figure 2(a) for the first 1,000 samples. The horizontal axis represents time and each time instance corresponds to a training sample. The exponentially decreasing lines represent the value of the update factor $\eta$ and the small circles depict actual updates of the weight vector $w$. The sign of the gain represents the class of the samples (above zero: object, below: background). Vertical lines represent positive training samples, all samples between these lines are negative samples. As can be seen in the figure, most positive training samples cause updates of the decision boundary, while the number of updates caused by the negative samples is much lower, implying that most information is contained in the positive training samples. The behavior of the bias is shown in Figure 2(b) for the complete training set, where it can be seen that it converges after a few thousand iterations (training samples).

From the results in Table 1, it can be seen that SGD converges in a single epoch. To evaluate how fast SGD converges towards the optimal solution, we have expanded the number of training samples step by step with 10\% at each iteration (after random shuffling) and have measured the obtained classification performance. Results are depicted in Figure 3. It is highly interesting to see that after learning only 20–30\% of the total training set, the curve is already partly coinciding with the optimal curve. Figure 3(b) shows the actual AUC which is fully in line with the previous conclusion, showing that already after 40\%, the final level is nearly reached.
5. Conclusions

We have incorporated the Stochastic Gradient Descent (SGD) algorithm for learning a linear SVM classifier in an object detection framework. Results on the challenging PASCAL 2006 Car dataset show that by incorporating SGD in the optimization process, only a single sweep over the training set is required. The obtained classification performance is similar to state-of-the-art SVM implementations, while obtaining a speedup factor in computation time of two to three orders of magnitude. Note that with increasing size of the training set, the speedup will even be larger due to the linear behavior of SGD.

A considerable benefit is that the classification performance quickly converges to the optimum, while only considering a part of the training set. Because the computation time of the optimization process is linearly scalable, this gives the user accurate control over the consumed execution times.

Because of its incremental behavior, SGD has the attractive feature that it enables online adaptation of the classification function and a classification model is available at any point in time. This enables solutions where training data arrives in a stream-like fashion where there is no time for a complete retraining of the classification model.

References