Benchmarking simple optimisation rules upon several metrics in remote sensing semantic segmentation

Adrien Chan-Hon-Tong

Benchmarking simple optimisation rules upon several metrics in remote sensing semantic segmentation

Adrien CHAN-HON-TONG

December 7, 2016

Abstract

In this paper, we focus on simple updating rules for linear separator optimisation in semantic segmentation context. We benchmark several rules upon several metrics. Our preliminary results show the relevancy of non classic rules on specialized metric.

1 Introduction

Object detection in image is well studied computer vision problem (see figure 1).

Recently, deep learning techniques [4, 12] have boosted object detection performances. [4] use a region proposal to transfer deep learning classification technique to object detection. While [12] tries to directly predict bounding boxes.

However, both box proposal and direct prediction of boxes have to deal with unstructured output: the number of boxes is data dependant and values in box encoding have different semantic (offset or size).

Thus, in this paper, we focus on semantic segmentation based detection. Semantic segmentation (see 2) is the goal of producing semantic mask of an images. Or, with others words, the goal of deciding a semantic label for each pixel of an image.

Optimizing deep learning is straightforward in semantic segmentation: output is a highly structured image with predefined size and a number of channels corresponding to the number of type of objects, which can be easily matched against the manual annotation.

So, we focus on semantic segmentation based detection, and, precisely, we are interested in this work by benchmarking simple update rules (see section 2 and 3) for learning the last layer parameters of such network (in other word, we are interested by selected a well designed loss for learning such network).

In this preliminary paper, we will not evaluated these rules under the final detection metric. Such results would anyway be biased by the selected pipeline for computing the bounding boxes from the semantic

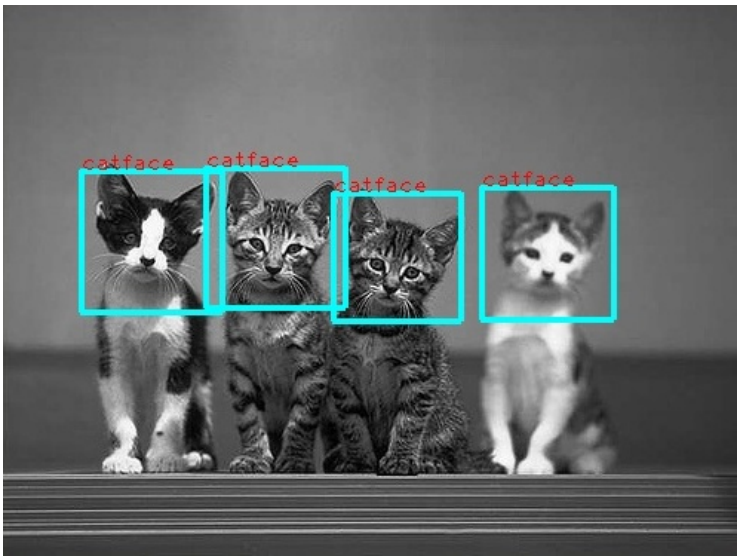
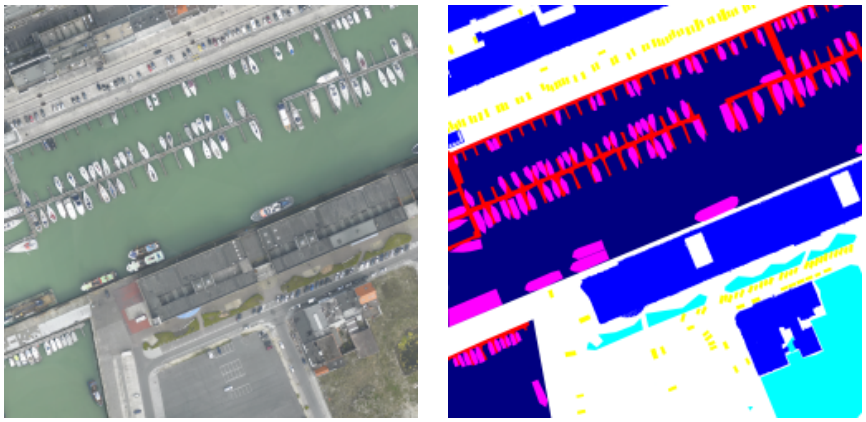


Figure 1: Illustration of object detection: putting bounding boxes on object in the image



(a) a remote sensing images (b) a semantic mask

Figure 2: Example of semantic segmentation of an image
Each color in the semantic mask is expected to correspond to a kind of object in the image. For example, boats in the image are in pink in the semantic mask while cars are yellow.

mask. Instead, we will benchmark these rules under several metric for semantic segmentation.

Considering several metrics in section 4 is relevant as the purpose of the semantic mask is to be a good middle output for final bounding box extraction. For example, maximizing the accuracy (ratio of well classified pixel over total pixel number) may prefer a segmentation of all pixels from 1 object over 2 than a partial segmentation of all objects which would definitely behave better for detection.

The current benchmark, we have obtained on the 2015 IEEE GRSS Data Fusion Contest (ground truth is provided by [9]) is presented in section 5 before current conclusion in section 6.

2 Anchor selection in SVM context

Let the function $relu$ verifies: $\forall u,\ relu\,(u) = \max\,(0, x)$.
Given $x_1, ..., x_N \in \mathbb{R}^D$ and $y_1, ..., y_N \in \{-1, 1\}$, the task of computing

$$\min_{w,b}\ ww + C\sum_n relu\,(1 - y_n\,(x_n w + b))$$

is the very well known C-SVM problem [3].

Middle size (for both N and D) instances can be efficiently solved with simple gradient descent [2] (small size can be solved exactly by linear programming discarding the ww term or be arbitrary approximated by designed algorithms [6]).

However, large size instances can not be solved this way. large scale solvers include stochastic gradient descent [1], cutting plane algorithm [8], online rules [13].

Except for cutting plane algorithm, these approaches are schematically built on the following structure: given $x_1, ..., x_N \in \mathbb{R}^D$ and $y_1, ..., y_N \in \{-1, 1\}$, select σ a permutation of $1, ..., N$ regarding w, b and update current solution regarding only $x_{\sigma(1)}, ..., x_{\sigma(R)}$ with $R \ll N$ and iterate.

In the following, we will call *anchors* the vectors selected during an iteration. Stochastic gradient descent, for instance, consists in selecting anchors uniformly and applying a standard gradient descent.

Now, what about others than uniform process to select anchors ? To resolve the C-SVM problem, non uniform prior may just lack theoretic guarantees compare to uniform one.

The interesting question is *what about others than uniform prior to select anchors if we do not exactly want to solve a C-SVM problem* ?

As a matter of fact, the standard C-SVM formula as several drawback regarding the context. In presence of very unbalanced population (e.g. $\sum_n y_n \ll 0$), standard C-SVM is known to badly behave as always predicting -1 ($w = 0, b = -1$) will be an almost perfect solution. For this problem both theoretic works (like [11]) on the correct energy and trick base patch are offered by the literature: sub sampling negative samples, over sampling positive ones, use two C one for negative and one for positive (see [5] for comparison of these threes) or bootstrapping [10].

Bootstrapping is especially relevant for the previous question as bootstrapping is exactly a non uniform process to select anchor. More precisely, bootstrapping consist to select hard examples (especially negative ones): examples which are the most wrongly classified.

Also, in segmentation context, [7] shows the relevancy of decreasing the weight given to samples close to a boundary of the ground truth segmentation. Pixel close to a boundary are likely to be hard example but maybe not for the correct reason: pixel close to boundary are likely to be wrongly annotated (see figure 2: manual annotation is coarse due to the difficulty to realize it). So, it is relevant not to focus too much on these pixels.

This can seem to be the opposite of bootstrapping, but it is more complicated.

Typically, when a segmented blob is completely wrongly classified then even far from boundary pixel are hard to classify. Such pixels far than boundary and wrongly classified are both hard example and safe example while wrongly classified border pixels are hard example but unsafe example. In other words, in [7], there are 2 distinct spaces: image space where pixel have proximity and pixel feature coding space. Focusing on different label vectors close in feature space is relevant but no focusing on different label vectors close in the image space is also relevant as the difference of label may be due to the label noise in the image space.

As we can see, non uniform prior to select anchors may be relevant in several case where C-SVM is not exactly what we want to achieve. This is especially true in detection/segmentation context which is the context of this work. Building on these previous works, we have benchmarked several anchor selections (deep learning compliant) and found that one selection interesting in detection/segmentation context.

3 Benchmarking anchor selections

Let $x_1, ..., x_N \in \mathbb{R}^D$ and $y_1, ..., y_N \in \{-1, 1\}$.
In this work, a anchor based gradient descent is a loop built on the following steps:

1. select σ permutation of $1, ..., N$ based on the current $w \in \mathbb{R}^D$, $b \in \mathbb{R}$
2. $\delta b = -\sum_r y_{\sigma(r)} relu'\left(1 - y_{\sigma(r)}\left(x_{\sigma(r)}w + b\right)\right)$
3. $\delta w = -\sum_r y_{\sigma(r)} x_{\sigma(r)} relu'\left(1 - y_{\sigma(r)}\left(x_{\sigma(r)}w + b\right)\right)$
4. $w = w - \alpha\delta w$, $b = b - \alpha\delta b$
5. go to 1

Technically, $relu$ is not a derivable function but anyway, we can arbitrarily chose that $relu'(0) = 1$. It may break optimal convergence properties of the algorithm but we do not really care about that as we may tackle non convex problem. Also, σ do not have to be a complete permutation: only the first R values are interesting.

Our main contribution is to benchmark a large set of anchor selections i.e. algorithm to select sub samples of the set of vectors. Let remark that

our goal is **NOT** only to solve the standard C-SVM (or even 2C-SVM), thus, one anchor selection can be particularly relevant for some metric while being bad for others.

In the following, we list the considered anchor selections.

Uniform sample: Select σ a random permutation.

bootstrapping: Bootstrapping consists in the following step

1. let $x_1, ..., x_N$ $y_1, ..., y_N$ be the input and w, b the current solution
2. let $p_1, ..., p_{t_p}$ be the subset of positive active elements of $x_1, ..., x_N$ i.e. $\exists i, j, x_i = p_j \Rightarrow y_i = 1$ and $wx_i + b \leq 1$. In addition, we sort this subset regarding w, b such that $wp_1 + b \leq ... \leq wp_{t_p} + b \leq 1$
3. let $n_1, ...n_{t_n}$ be the corresponding subset for the active negative elements of $x_1, ..., x_N$: $\exists i, j, x_i = n_j \Rightarrow y_i = -1$ and $-1 \leq wn_1 + b \leq ... \leq wn_{t_n} + b$
4. that bootstrapping consists in selecting hard positive and negative elements i.e. $\sigma(2i) = p_i$ and $\sigma(2i+1) = n_{t_n - i}$ (this may not lead to a permutation but it is an acceptable definition of the first R values assuming $2R \leq t_p, 2R \leq t_n$

There exists multiples implementations of bootstrapping: selecting hard positives and negatives or selecting hard negatives + random positives... In the following, we chose to select 1/4 hard positives, 1/4 hard negatives, 1/4 random positives, 1/4 random negatives.

precision-first: We will call in the following precision-first the mutation of bootstrapping where we select hard active negatives and easy active positives in order to give more importance to precision than to recall i.e. $\sigma(2i) = p_{t_p - i}$ and $\sigma(2i+1) = n_{t_n - i}$ with previous definitions of $p_1, ..., p_{t_p}$ and $n_1, ..., n_{t_n}$.

bootstrapping without boundary: This anchor selection inspired from [7] consists in performing classic bootstrapping with exclusion of all vectors x_i corresponding to a pixel close to a boundary of the manual segmentation.

Here, we introduce the process generating the vector $x_1, ..., x_N$ which in our case is a segmentation problem. Given an image of size $W \times H \times 3$ (3 for a color images) firstly a feature mapping is performed onto the image leading to a $W \times H \times D$ image (assuming there is no pooling in the feature mapping). Then, each $W \times H$ pixels is a D dimensional vector associated to a semantic label (which is -1 or 1 for binary segmentation which is the only handled segmentation in this preliminary version of the paper).

Thus, only one VGA image (640x480) leads to a cloud of 307200 vectors of dimension D. This last remark highlights the need for selecting anchor in the gradient descent as semantic segmentation database may contains several thousand such images.

precision-first per component: This last anchor selection is one of the contribution of this paper as we do not found it on the literature. It consists in selecting anchors taking into account not only $wx + b$ but also the geometric origin of the pixel.

More precisely, the idea is to select a fix number of pixel per positive connected component.

For example, assuming we have only *boat* and *no boat* classes, this process consists in selecting let say 4 pixels from each pink blob of the figure 2 (independently, from the size of the blob or from it context).

formally, let $p_{1,1}, ..., p_{1,t_1}, p_{2,1}, ..., p_{2,t_2}, \ ..., \ p_{L,1}, ..., p_{L,t_L}$ be all the active positive examples sorted but clustered by connected component (all $p_{l,1}, ..., p_{l,t_l}$ belong to a same connected component in the segmentation mask). Let $n_1, ..., n_T$ be the sorted active negative like before.

Then, if $R = 2L$ $\sigma(l) = p_{t_l}$ for $l \leq L$ and $\sigma(l) = n_{T-l}$ for $l > L$.

Like precision first, this one will try to focus on easy positives but avoiding taking all the easy positives from a same blob. Instead, it forces to consider several easy positives from each bloc.

4 Metrics

As segmentation is just a middle features in our purpose, the exact pixelwise segmentation is not what we are trying to reach ; but we do not directly perform detection with bounding boxes as bounding boxes are an unstructured data structure harder to model with CNN.

So, we rely on different metric to benchmark the different anchor selection.

- number of false alarm vs number of correct positives
- number of false alarm vs number of correct positive connected component: here the objective is only to have one positive pixel per positive connected component. Of course, this metric will largely advantage *precision-first per component* anchor selection but on the other hand it is what we need a minima to perform detection after: at least one clue per object

5 Results

We choose to use the 2015 IEEE GRSS Data Fusion Contest (DFC) to perform our experiments. This dataset (called grss_dfc_2015 in this article) is composed of 6 large size (10000x10000 pixels) remote sensing ortho-images with resolution of around 5cm (only RGB image are used in this experiment).

A manual semantic labelling has been made by [9] with 7 classes (we only keep *car* and *no car* to reduce the problem to a binary one).

We split the dataset in 2 sets train/test of 4/2 images.

We apply a VGG16 upon layer conv3_3 with a terminal 9x9 convolution without finetuning for feature extraction resulting into a 4 pooling. Dimension of the vectors is thus $20736 = 9 \times 9 \times 256$.

anchor selection	false alarm	accuracy	accuracy component
uniform	3973537	74%	95%
bootstrap	1173787	54%	95%
precision first	306	5%	14%
bootstrap without border	1053244	59%	95%
precision first per comp	1854	7%	67%

Table 1: Results of the benchmark

The results of the benchmark (in this preliminary version, the benchmark is just done performed on a subset of the DFC) are given in table 1.

6 Partial conclusion

Even if our benchmarking is currently done on a small amount of data, we think that the current results have interesting features: a careful design of the anchor selections have a dramatic and non uniform influence over the results of the different metrics.

This invite to strengthen our current benchmark as the persistence of the current features will be an interesting knowledge for the community.

References

[1] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[2] Olivier Chapelle. Training a support vector machine in the primal. *Neural computation*, 19(5):1155–1178, 2007.

[3] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[4] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition*, 2014.

[5] He He and Ali Ghodsi. Rare class classification by support vector machine. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 548–551. IEEE, 2010.

[6] Don Hush and Clint Scovel. Polynomial-time decomposition algorithms for support vector machines. *Machine Learning*, 51(1):51–71, 2003.

[7] Mingyuan Jiu, Christian Wolf, Graham Taylor, and Atilla Baskurt. Human body part estimation from depth images via spatially-constrained deep learning. *Pattern Recognition Letters*, 50:122–129, 2014.

[8] T. Joachims, T. Finley, and Chun-Nam J. Yu. Cutting-plane training of structural svms. *Machine Learning*, 77(1):27–59, 2009.

[9] Adrien Lagrange, Bertrand Le Saux, Anne Beaupere, Alexandre Boulch, Adrien Chan-Hon-Tong, Stéphane Herbin, Hicham Randrianarivo, and Marin Ferecatu. Benchmarking classification of earth-observation data: from learning explicit features to convolutional networks. In *International Geoscience and Remote Sensing Symposium*, 2015.

[10] Constantine P Papageorgiou, Michael Oren, and Tomaso Poggio. A general framework for object detection. In *Computer vision, 1998. sixth international conference on*, pages 555–562. IEEE, 1998.

[11] Shameem Puthiya Parambath, Nicolas Usunier, and Yves Grandvalet. Optimizing f-measures by cost-sensitive classification. In *Advances in Neural Information Processing Systems*, pages 2123–2131, 2014.

[12] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, 2015.

[13] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, 127(1):3–30, 2011.