Self-Paced Learning for Semisupervised Image Classification

Kevin Miller
Stanford University
Palo Alto, CA
kjmiller@stanford.edu

Abstract

In this project, we apply three variants of self-paced learning to semi-supervised image classification with latent bounding boxes. Intuitively, these three variants aim to find better local optima than the baseline algorithm (LSSVM) by ignoring "difficult" data during the earlier iterations of the algorithm. Indeed, lower objectives are achieved by all three variants, but they do not translate into better average precision scores than those achieved by the LSSVM baseline.

1. Introduction

In this project, we aim to train a classifier to predict which objects an image contains. Specifically, during training we are given a set X containing N images, and for each image $x_i \in X$, we are given a set of object labels $Y_i \subset Y$ (where Y is the set of all possible object labels) indicating which objects are contained in the image x_i. In this particular experiment, Y_i is always non-empty and is always a strict subset of Y.

Theoretically it would be possible to train a non-binary classifier to predict each Y_i by representing each possible Y_i as a possible classification. However, this would require us to deal with $2^{|Y|}$ different possible classifications, which could lead to tractability and data sparsity issues. Therefore, we approach this problem by training a traditional non-binary classifier (i.e. the type one would typically use in the simpler problem in which each image contained just one type of object) that gives a classification score to each member of Y for each image. These classification scores can then be used to compute an average precision score for each member of Y. The AP scores essentially tell us how well our classifier would do if we predicted each Y_i by thresholding the classifier scores.

Since we are given the ground truth during training, it would be possible to learn a classifier via a supervised learning; however, one can often learn much more powerful models by using latent variables that reduce noise. In this case, for each image x_i, the latent variable h_i represents the location and dimensions of a bounding box that ideally contains one of the objects in x_i. H_i is the set of all possible bounding boxes for image i.

2. Related Work

In [1], Felzenszwalb et. al. formulate the Latent SVM (LSVM) algorithm and use it to train a deformable parts model for semisupervised binary classification and detection. In [2], Yu, Joachims, and Finley put forward an extension of this algorithm, the Latent Structural SVM (LSSVM), which can efficiently learn models for classification problems with large prediction and latent variable spaces. Like the LSVM, the LSSVM problem is a non-convex problem, and although the optimization algorithm used in [2] can be shown to be a descent algorithm, it is still susceptible to potentially bad local minima.

In [3], Koller, Packer, and Kumar formulate an optimization algorithm for LSSVM called *Self-Paced Learning* (SPL), which allows examples to be ignored during the optimization of the model w but increasingly rewards the inclusion of examples. SPL is actually quite similar to an independently-derived, fully-supervised algorithm in [5] called *Robust SVM*, which attempts to detect and ignore outliers while training an SVM. Intuitively, SPL can be thought of as an algorithm that ignores "difficult" examples (such as outliers) during early outer iterations and only includes them later.

In [3], SPL is shown to outperform the standard optimization algorithm for LSSVM (which will be referred to simply as "LSSVM" for notational convenience) on a very simple semisupervised bounding box problem. In this project, we apply a slightly modified version of SPL to a more complicated, larger-scale bounding-box problem. In addition, in this project we formulate two extensions of the SPL algorithm and apply them to our bounding box problem.

3. Approach

3.1. Features and Preprocessing

Our feature vector function $\Psi(x, y, h)$ is a concatentation of K such subfunctions $\Psi_1(x, y, h), ..., \Psi_K(x, y, h)$, referred to as feature sets or "kernels". The kernels we use are HoG, SIFT, color-SIFT, RGB-SIFT, and opponent-SIFT. What makes this set of kernels interesting is the fact that HoG and SIFT operate on greyscale versions of the images, while color-SIFT, RGB-SIFT, and opponent-SIFT are extensions of SIFT into the color domain (as described in [4]); intuitively, we might expect HoG and SIFT to be considered "easier" features for some types of images where color isn't well-corellated with object type. We form each $\Psi_k(x, y, h)$ by using a codebook to discretize the descriptors and then forming a 2-level spatial pyramid (via BoW max-pooling) over the part of the image within the bounding box specified by h and concatenating it with a BoW max-pooling over the descriptors that fall outside of the bounding box. Thus, our model takes into account information both inside and outside of the bounding box, as well as information in each quadrant of the bounding box. Each $Psi_K(x, y, h)$ can be thought of as having $|Y|$ subsections, each with a 1 entry at the beginning to allow for bias terms; y simply determines which of these subsections has the actual feature data - the rest of the subsections are zeroed out. In effect, this means that $w^T\Psi(x, y, h)$ is simply the inner product of one subsection of w (indexed by y) with the actual feature data.

The set of possible bounding boxes for each image is determined by computing objectness scores for a large range of potential bounding boxes and then choosing the bounding boxes with the top 50 objectness scores.

3.2. Algorithms

In order to train a classifier that will get good AP scores, we construct an *expanded training set* as follows:

For each $x_i \in X$, duplicate $|Y_i|$ times, and for each duplicate x_j, let $y_j \in Y_i$ be the "true" label and let all other labels in Y_i be "whitelist" labels that aren't used in slack constraints (call set of whitelist labels $\tilde{Y}_i$.

Our baseline algorithm (referred to as LSSVM, despite minor differences with the usual LSSVM algorithm) is as follows:

$w \leftarrow 0$
Randomly initialize $h_1^*, ..., h_{\tilde{N}}^*$
while Objective not yet converged **do**
  $w \leftarrow \min_{\{} w, \xi\} \frac{1}{2}||w||_2^2 + \frac{C}{\tilde{N}} \sum_{i=1}^{\tilde{N}} \xi_i$
  subject to,
  $w^T\Psi(x_i, \bar{y}, \bar{h}) - w^T\Psi(x_i, y_i, h_i^*) + \Delta(y_i, \bar{y}) \leq \xi_i \forall (\bar{h}, \bar{y}) \in H_i \times (Y - \tilde{Y}_i)$
  for $i = 1 \rightarrow \tilde{N}$ **do**
    $h_i^* \leftarrow \arg\max_{h \in H_i} w^T\Psi(x_i, y_i, h)$
  end for
end while

The 1-slack cutting-plane algorithm described in [6] is used to optimize for w.
In our version SPL (which is similar to the one used in [3]), we add a constraint on how many images in $\tilde{X}(y)$ are included for each $y \in Y$ (where $\tilde{X}(y) = \{\tilde{x}_i \in \tilde{X} : y_i = y)\}$. The SPL algorithm is illustrated below:

$w \leftarrow 0$
Randomly initialize $h_1^*, ..., h_{\tilde{N}}^*$
$f \leftarrow f_0$
while Objective not yet converged **do**
  for $s = 1 \rightarrow S$ **do**
    $v \leftarrow \arg\min_{\{} v, \xi\} \frac{C}{\tilde{N}} \sum_{i=1}^{\tilde{N}} v_i \xi_i$
    subject to,
    $w^T \Psi(x_i, \bar{y}, \bar{h}) - w^T \Psi(x_i, y_i, h_i^*) + \Delta(y_i, \bar{y}) \leq \xi_i \forall (\bar{h}, \bar{y}) \in H_i \times (Y - Y_i)$
    $\sum_{i:x_i \in \tilde{X}(y)} v_i == \lceil f|\tilde{X}(y)| \rceil \forall y \in Y$
    $w \leftarrow \arg\min_{\{} w, \xi\} \frac{1}{2}||w||_2^2 + \frac{C}{\tilde{N}} \sum_{i=1}^{\tilde{N}} v_i \xi_i$
    subject to,
    $w^T \Psi(x_i, \bar{y}, \bar{h}) - w^T \Psi(x_i, y_i, h_i^*) + \Delta(y_i, \bar{y}) \leq \xi_i \forall (\bar{h}, \bar{y}) \in H_i \times (Y - Y_i)$
  end for
  for $i = 1 \rightarrow \tilde{N}$ **do**
    $h_i^* \leftarrow \arg\max_{h \in H_i} w^T \Psi(x_i, y_i, h)$
  end for
  $f \leftarrow \min\{f + \Delta f, 1\}$
end while

In an extension referred to as *SPL+*, we can include or ignore (image, kernel) pairs rather than entire images. The algorithm works as follows:

$w \leftarrow 0$
Randomly initialize $h_1^*, ..., h_{\tilde{N}}^*$
$f \leftarrow f_0$
while Objective not yet converged **do**
  for $s = 1 \rightarrow S$ **do**
    $v \leftarrow \arg\min_{\{} v, \xi\} \frac{C}{\tilde{N}} \sum_{i=1}^{\tilde{N}} \xi_i$
    subject to,
    $\sum_{k=1}^{K} (v_i^{(k)} w_k^T \Psi_k(x_i, \bar{y}, \bar{h}) - v_i^{(k)} w_k^T \Psi_k(x_i, y_i, h_i^*) + \frac{v_i^{(k)}}{K} \Delta(y_i, \bar{y})) \leq \xi_i \forall (\bar{h}, \bar{y}) \in H_i \times (Y - \tilde{Y}_i)$
    $\sum_{i:x_i \in \tilde{X}(y)} v_i^{(k)} == \lceil f|\tilde{X}(y)| \rceil \forall y \in Y, k \in \{1, ..., K\}$
    $w \leftarrow \arg\min_{\{} w, \xi\} \frac{1}{2}||w||_2^2 + \frac{C}{\tilde{N}} \sum_{i=1}^{\tilde{N}} \xi_i$
    subject to,
    $\sum_{k=1}^{K} (v_i^{(k)} w_k^T \Psi_k(x_i, \bar{y}, \bar{h}) - v_i^{(k)} w_k^T \Psi_k(x_i, y_i, h_i^*) + \frac{v_i^{(k)}}{K} \Delta(y_i, \bar{y})) \leq \xi_i \forall (\bar{h}, \bar{y}) \in H_i \times (Y - \tilde{Y}_i)$
  end for
  for $i = 1 \rightarrow \tilde{N}$ **do**
    $h_i^* \leftarrow \arg\max_{h \in H_i} w^T \Psi(x_i, y_i, h)$
  end for
  $f \leftarrow \min\{f + \Delta f, 1\}$
end while

v is optimized as follows:

Randomly initialize v such that
for $y \in Y$ **do**
  for $t = 1 \rightarrow T$ **do**
    for $k = 1 \rightarrow K$ **do**
      Greedily optimize $v_i^{(k)} \forall i$ such that $x_i \in \tilde{X}(y)$
    end for
  end for
end for

This update step admits two levels of parallelization; first, the for loop across $y \in Y$ can be parallelized, and second, the greedy update step can be parallelized across $i : x_i \in \tilde{X}(y)$.
In a further extension referred to as *SPL++*, we can now include or ignore (image, kernel, label) 3-tuples, where "label" refers to the $\bar{y}$ that we consider in the slack constraint for the w optimization problem. Intuitively, SPL++ should allow us to penalize instances where, for example, our model would misclassify a cat as a car but not penalize instances where it would misclassify a cat as a dog. The SPL++ algorithm works as follows:

$w \leftarrow 0$
Randomly initialize $h_1^*, ..., h_{\tilde{N}}^*$
$f \leftarrow f_0$
while Objective not yet converged **do**
  for $s = 1 \rightarrow S$ **do**
    $v \leftarrow \arg\min_{\{} v, \xi\} \frac{C}{\tilde{N}} \sum_{i=1}^{\tilde{N}} \xi_i$
    subject to,

$\sum_{k=1}^{K}(v_i^{(k,\bar{y})} w_k^T \Psi_k(x_i, \bar{y}, \bar{h}) - v_i^{(k,\bar{y})} w_k^T \Psi_k(x_i, y_i, h_i^*) + \frac{v_i^{(k,\bar{y})}}{K}\Delta(y_i, \bar{y})) \leq \xi_i \forall (\bar{h}, \bar{y}) \in H_i \times (Y - \tilde{Y}_i)$
$\sum_{i:x_i \in \tilde{X}(y)} v_i^{(k,\bar{y})} == \lceil f|\tilde{X}(y)| \rceil \forall y \in Y, k \in \{1, ..., K\}, \bar{y} \in Y$ (note that images for which $\bar{y} \in \tilde{Y}_i$ are not counted in this constraint)
$w \leftarrow \arg\min_{\{w,\xi\}} \frac{1}{2}||w||_2^2 + \frac{C}{\tilde{N}} \sum_{i=1}^{\tilde{N}} \xi_i$
subject to,
$\sum_{k=1}^{K}(v_i^{(k,\bar{y})} w_k^T \Psi_k(x_i, \bar{y}, \bar{h}) - v_i^{(k,\bar{y})} w_k^T \Psi_k(x_i, y_i, h_i^*) + \frac{v_i^{(k,\bar{y})}}{K}\Delta(y_i, \bar{y})) \leq \xi_i \forall (\bar{h}, \bar{y}) \in H_i \times (Y - \tilde{Y}_i)$
end for
for $i = 1 \rightarrow \tilde{N}$ **do**
$h_i^* \leftarrow \arg\max_{h \in H_i} w^T \Psi(x_i, y_i, h)$
end for
$f \leftarrow \min\{f + \Delta f, 1\}$
end while

The optimization of v for SPL++ is analogous to the optimization for SPL+.
In practice, in order to obtain a good initial w and avoid bad local (w, v) minima, the first two outer iterations of SPL, SPL+, and SPL++ are LSSVM outer iterations.

4. Experiment

4.1. Experimental Setup

A subset of the training set for VOC 2007 is used. The total size of this subset is 800 images. We use 4-fold cross-validation. Also, we only consider the 7 most frequent classes which are:
1. person
2. car
3. chair
4. dog
5. sofa
6. bird
7. cat

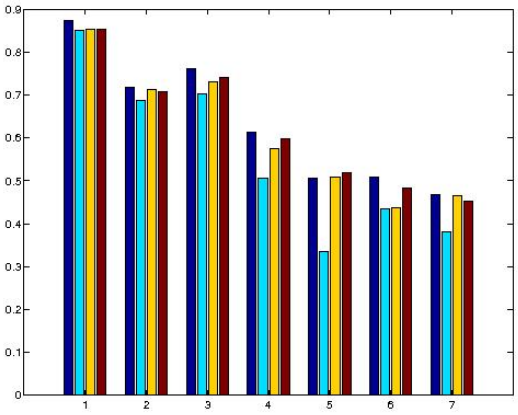
(This indexing scheme is used in plots that involve different classes).

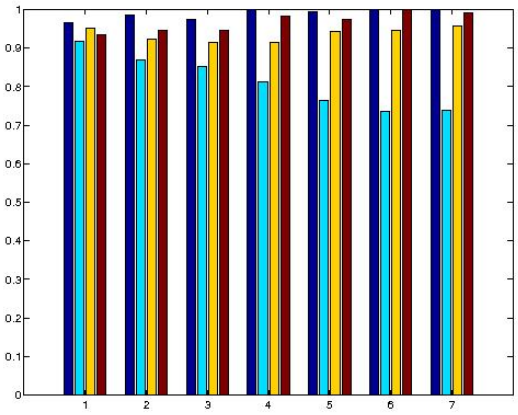
4.2. Preliminary Work

A few different values of C were tried at first, but the performace of LSSVM did not seem to vary much, so we chose to go forward with $C = 1.0$. Also, after some initial poor results from SPL and SPL+, we decided to reweight the slacks of each example so that it would be as if we were training on a balanced data set. This did improve the performance of LSSVM somewhat. Full runs were thus subsequently done on the reweighted dataset with $C = 1.0$.

4.3. Results

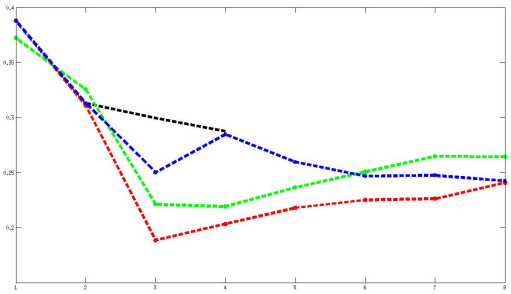
Below are the average test and train AP scores of LSSVM, SPL, SPL+, SPL++ on each class (within each cluster in the bar graphs below, the ordering is LSSVM, SPL, SPL+, SPL++).





Thus, there was only one class for which LSSVM

did not have the best performance of any of the algorithms. This is somewhat strange, given that SPL, SPL+, and SPL++ consistently achieve a lower objective than LSSVM, as shown below for fold 3 (where black is LSSVM, red is SPL, green is SPL+, and blue is SPL++):

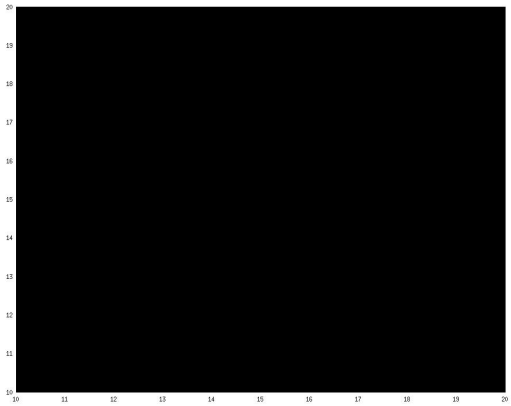


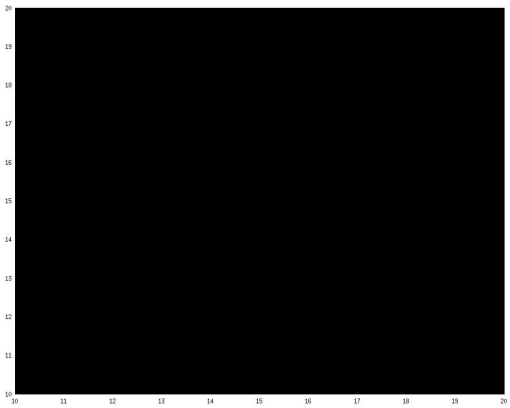
This, combined with the fact that SPL, SPL+ and SPL++ significantly underperform on train AP relative to LSSVM, leads us to suspect that SPL, SPL+, and SPL++ are leading to local minima that are lower than what LSSVM reaches but more underfitting to the data. It makes sense that SPL, SPL+, and SPL++ would lead to an underfitting local minimum, given that they ignore data that would lead to high slack values (and likely overfitting).

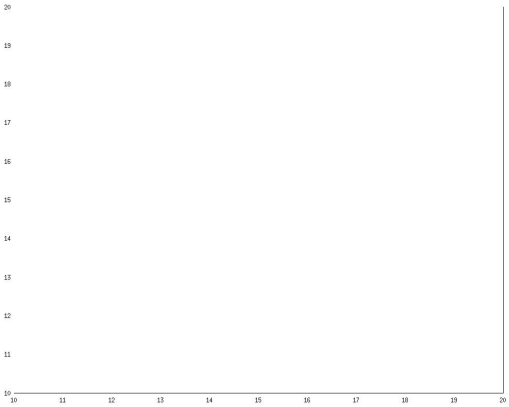
4.4. Close Analysis of Self-Pacing

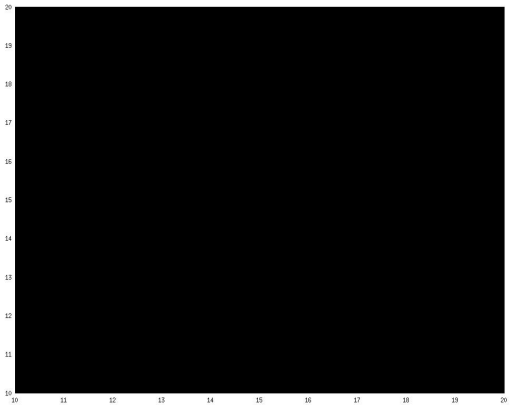
In the following instances, heatmaps are used to represent v; black represents inclusion, while white represents exclusion.

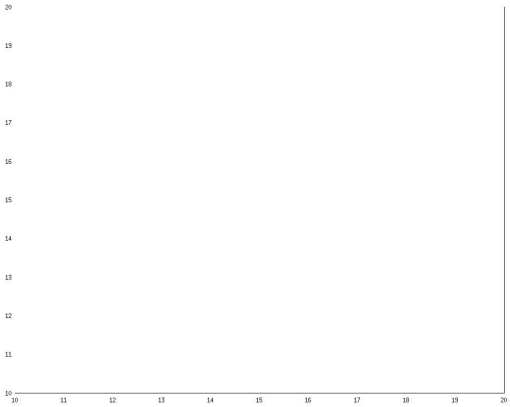
4.4.1 SPL

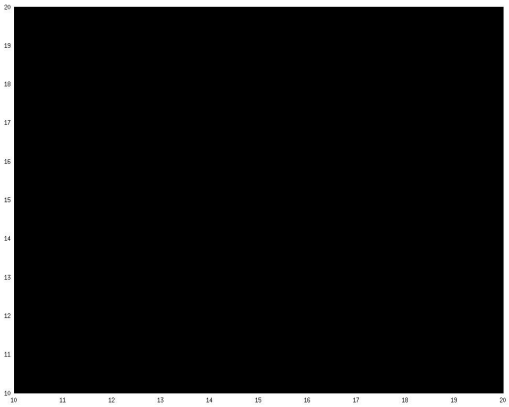


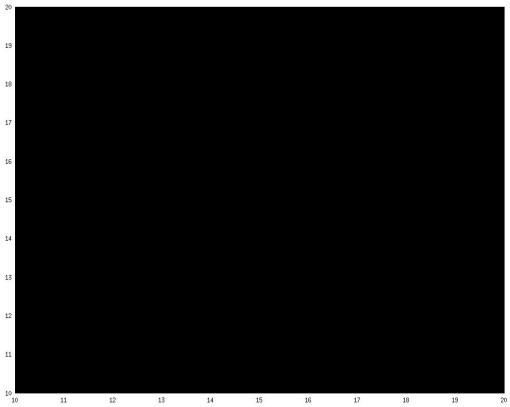






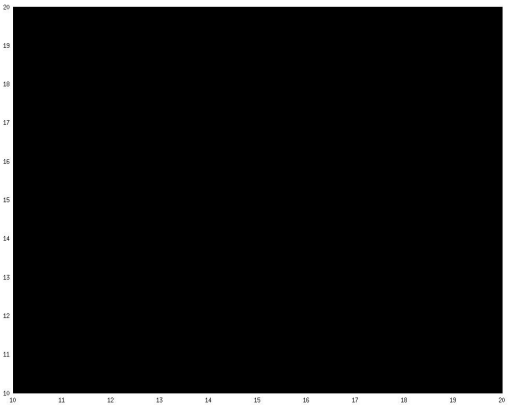


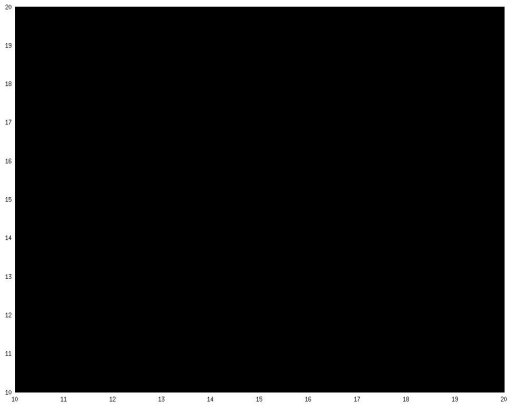


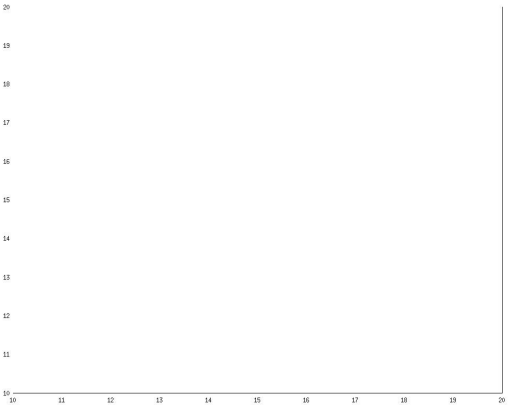


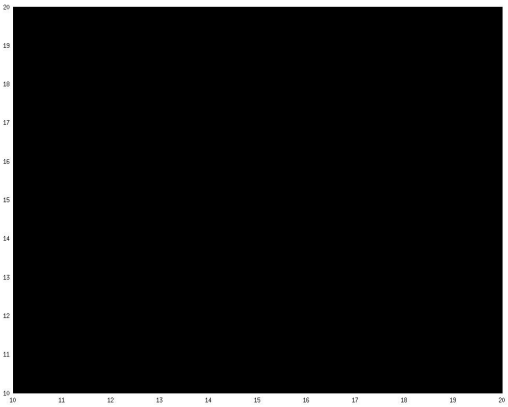
The temporary exclusion of this image coincides with the widening of the bounding box. Perhaps this widened bounding box is the result of an underfitting of w. It's not surprising that this image is considered difficult at some point - there are many people in it, all of them wiht different poses, making it difficult to get a bounding box over most of just one "typical" person. A wider bounding box would throw away more information within itself because more points inside of it would get max-pooled together.

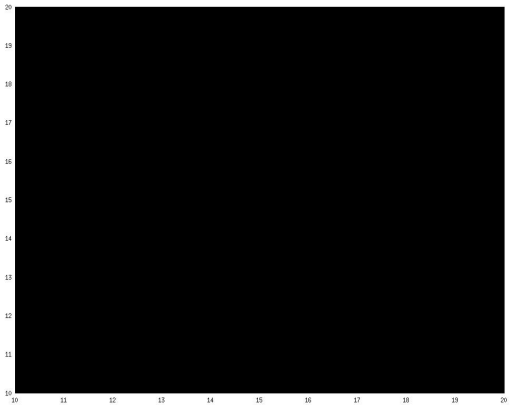
Another interesting illustration of SPL's behavior can be seen in the following example:

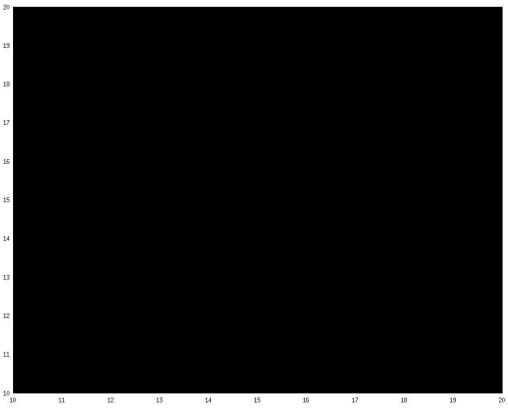

20
19
18
17
16
15
14
13
12
11
10
10 11 12 13 14 15 16 17 18 19 20

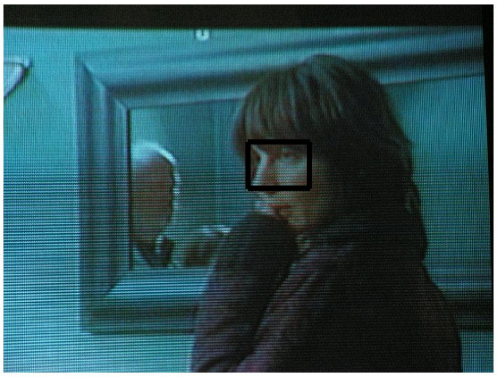

20
19
18
17
16
15
14
13
12
11
10
10 11 12 13 14 15 16 17 18 19 20

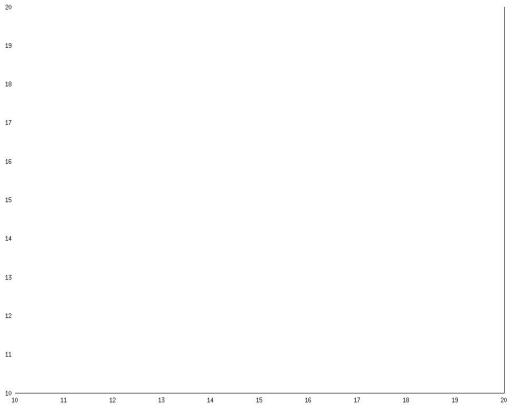

20
19
18
17
16
15
14
13
12
11
10
10 11 12 13 14 15 16 17 18 19 20

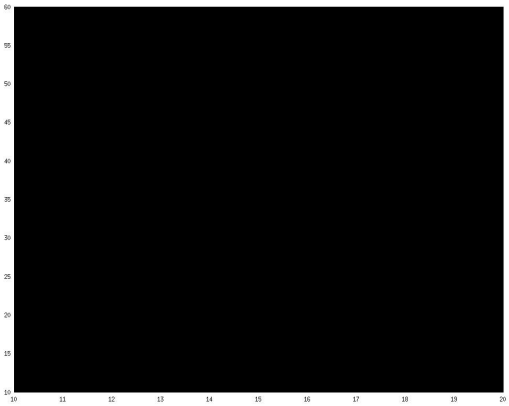

10
11
12
13
14
15
16
17
18
19
20

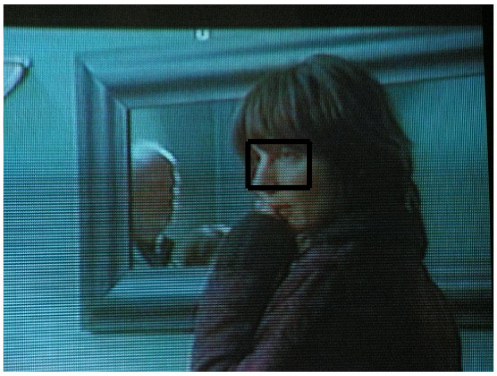

10
11
12
13
14
15
16
17
18
19
20


10
11
12
13
14
15
16
17
18
19
20





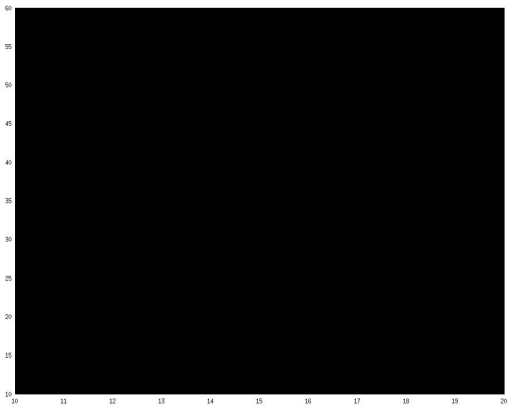


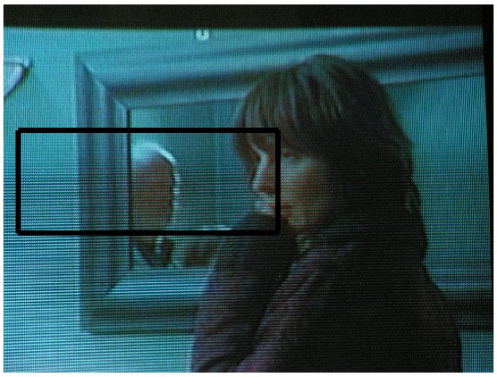


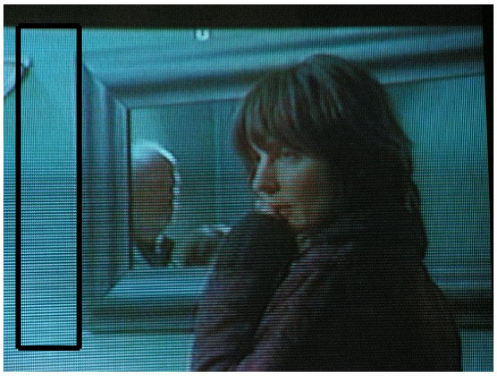
At first, it seems, the algorithm tries to classify the child as a person but then decides that the child is too difficult to classify as a person, at which point the bounding box moves toward (and ends up surrounding the top of) the couch. In the LSSVM algorithm, by contrast, the bounding box always surrounds the child, from beginning to end. Perhaps if one looked at more images, one might find a general pattern of SPL discarding ”difficult” people examples and underfitting its model of people.

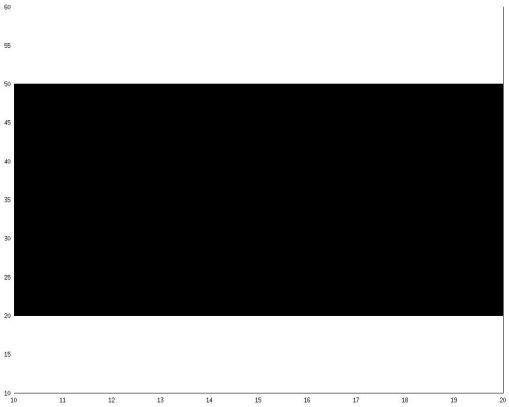
4.4.2 SPL+

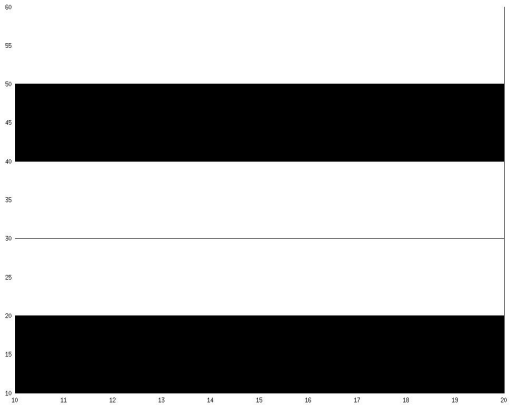
The rows, going upwards, represent HoG, SIFT, color-SIFT, RGB-SIFT, and opponent-SIFT.

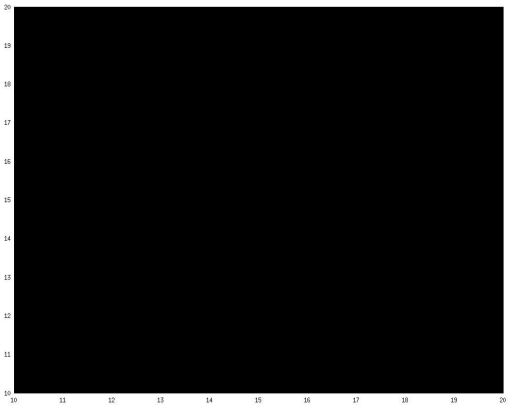

60
55
50
45
40
35
30
25
20
15
10
10
11
12
13
14
15
16
17
18
19
20





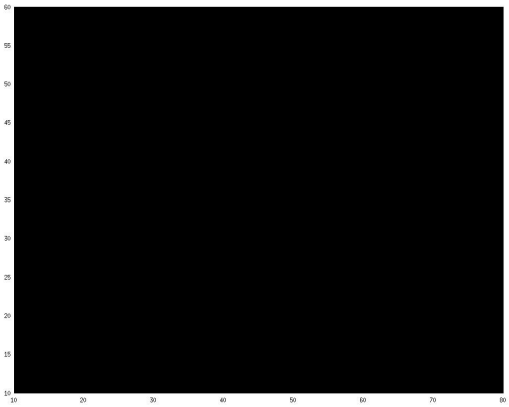

60
55
50
45
40
35
30
25
20
15
10
10
11
12
13
14
15
16
17
18
19
20

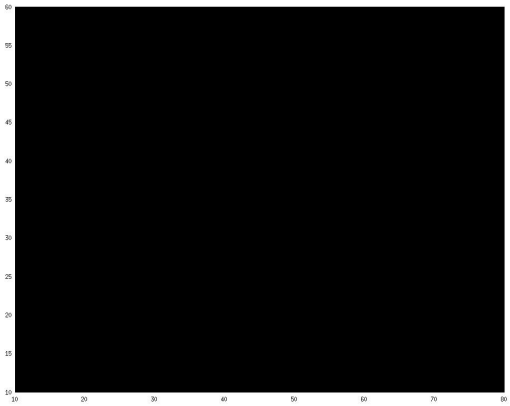

60
55
50
45
40
35
30
25
20
15
10
10
11
12
13
14
15
16
17
18
19
20

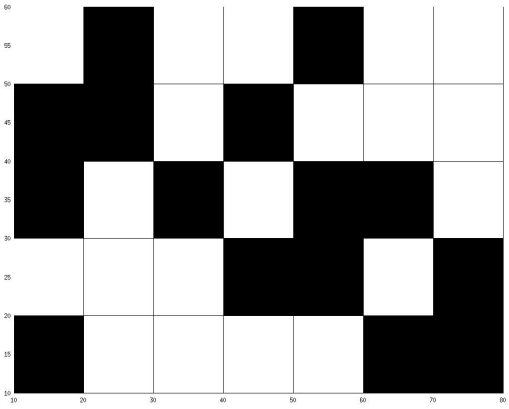


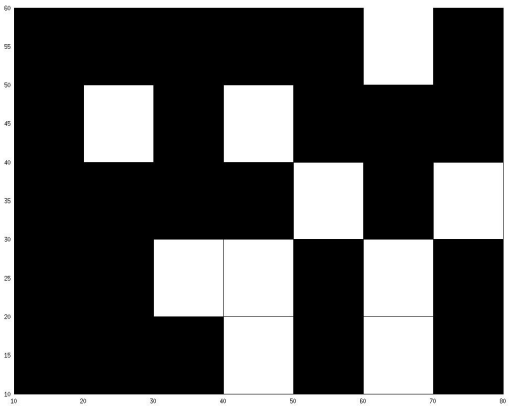
The deviation of the bounding box from the woman's face seems to coincide with the ignoring of one or more of the color-sensitive kernels. It makes sense that color-sensitive kernels would be ignored for this example, since its coloring is quite weird. It also makes sense that practically all of the kernels are ignored at some point - one would not expect the the black grid lines to make classification using orientation-based features any easier.

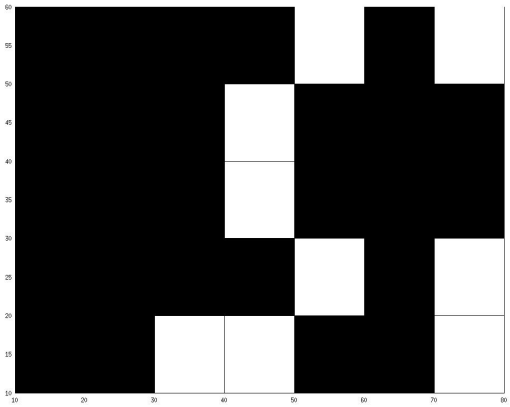
4.4.3 SPL++

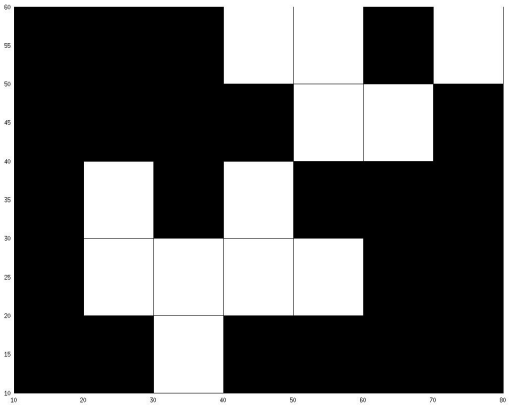


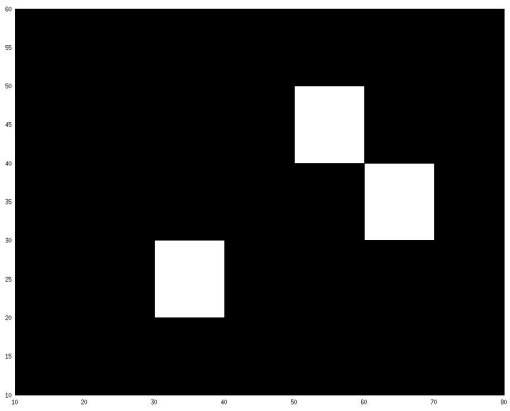












Here we can see that there are typically many variables ignored in the "dog" and "cat" columns (columns 4 and 7, respectively), which makes sense, since we would expect horses to be easy to confuse with cats and dogs (and horses aren't even an included class in our problem, making it even more difficult for our model to be trained to not call a horse a cat or a dog).

5. Conclusion

SPL, SPL+, and SPL++ in general do not outperform LSSVM but do show promise, due to their ability to achieve lower objective values. Based on this, a next step would be to try higher values of C, which would make underfitting less likely. Other good ideas might be to run the greedy update steps more thoroughly (in these experiments, both S and T were set to 1 due to time constraints). Finally, since our AP scores are intuitively really related to binary classification, a different reweighting of the examples that balanced images with a particular type of object against images without that type of object could potentially lead to AP scores that better reflected the objective values.

References

[1] P. Felzenszwalb, D. McAllester, and D. Ramanan A Discriminatively Trained, Multiscale, Deformable Part Model *CVPR*, 2008.

[2] T. Joachims and C.-N. Yu. Learning Structural SVMs with Latent Variables *ICML*, 2009

[3] M. Pawan Kumar, B. Packer, D. Koller Self-Paced Learning for Latent Variable Models *NIPS*, 2010.

[4] Koen E. A. van de Sande, Theo Gevers, and Cees G. M. Snoek Evaluating Color Descriptors for Object and Scene Recognition *IEEE Transactions on Pattern Analysis and Machine Intelligence* volume 32 (9), pages 1582-1596, 2010

[5] L. Xu, K. Crammer, D. Schuurmans Robust Support Vector Machine Training via Convex Outlier Ablation *AAAI* 2006

[6] T. Joachims, T. Finley, and C.-N. Yu. Cutting-plane training for structural SVMs *Machine Learning* 77(1):27-59, 2009

6. Appendix

This project is part of my research with Daphne Koller, Ben Packer, and M. Pawan Kumar. I have been collaborating with Rafi Witten, and undergraduate who is not on CS231A, on much of this research.

1. Explicitly explain what the computer vision components are in this course project;

2. Explicitly list out all of your own contributions in this project in terms of:

   (a) ideas Original SPL was formulated by Koller, Kumar, and Packer. They also came up with the main ideas behind SPL+ and SPL++. However, I collaborated with them and Rafi on exactly how to pose these ideas in the form of an optimization problem.

(b) formulations of algorithms I did most of the work in formulating the update steps for SPL+ and SPL++.

(c) software and coding Rafi and I collaborated closely on the main LSSVM algorithm. The SPL, SPL+ and SPL++ update code was entirely my own work, as was all the code at places where SPL, SPL+, and SPL++ interacted with the normal LSSVM algorithm. Feature computation and computation of objectness scores for bounding boxes were entirely Rafi's work, not mine.

(d) designs of experiments I collaborated with Rafi, as well as my advisors on choosing AP scores as a metric. However, design choices for the particular experiments for this project were entirely my own (we're currently trying to run much bigger experiments as a longer-term goal).

(e) analysis of experiments This was entirely my own work.

3. Verify and confirm that you (and your partner currently taking CS231A) are the sole author(s) of the writeup. Please provide papers, theses, or other documents related to this project so that we can compare with your own writeup. See CURIS presentation attached to submission email. I collaborated with Rafi on that last summer.