Conditional Random Fields for Object Localization

This thesis uses a modified version of the LaTeX style file for Foundations and Trends® from now Publishers, Inc. *www.nowpublishers.com*

Conditional Random Fields for Object Localization

Andreas Christian Eilschou

Department of Computer Science
University of Copenhagen
Denmark
jwb226@alumni.ku.dk

Andreas Hjortgaard Danielsen

Department of Computer Science
University of Copenhagen
Denmark
gxn961@alumni.ku.dk

Supervisors

Christian Igel (University of Copenhagen)
Christoph H. Lampert (IST Austria)

UNIVERSITY OF COPENHAGEN

Department of Computer Science
University of Copenhagen
Universitetsparken 1
2100 Copenhagen
Denmark
www.diku.dk

Institute of Science and Technology Austria
Am Campus 1
3400 Klosterneuburg
Austria
www.ist.ac.at

Conditional Random Fields for Object Localization

Andreas Christian Eilschou[1] and Andreas Hjortgaard Danielsen[2]

[1] *Department of Computer Science, University of Copenhagen, Denmark, jwb226@alumni.ku.dk*
[2] *Department of Computer Science, University of Copenhagen, Denmark, gxn961@alumni.ku.dk*

Abstract

We propose a probabilistic model of the object localization problem by formulating it in the framework of conditional random fields (CRFs) which allows parameter learning by maximum conditional likelihood (MCL) estimation. The true log-likelihood and its gradient are infeasible to compute even for medium-sized images and datasets and we consider four approximate methods for learning the parameters of the CRF model: stochastic gradient descent, contrastive divergence, pseudolikelihood and piecewise training.
Our contribution is threefold: First, we formulate and derive the true log-likelihood objective and its approximations for the object localization problem. Secondly, we show that the approximate learning methods obtain results comparable with exact MCL learning but only contrastive divergence, pseudolikelihood and piecewise training are feasible in practice. Thirdly, we show that our results are comparable with results obtained by a structured support vector machine.

Resumé

Vi foreslår en sandsynlighedsgrafisk model af objektlokaliseringsproblemet ved at formulere det som et conditional random field (CRF), hvilket tillader parameterlæring ved hjælp af maximum conditional likelihood (MCL) estimering. Den sande log-likelihood og dens gradient er uoverkommelige at beregne, selv for middelstore billeder og datasæt, og vi tager fire approksimationsmetoder i betragtning til indlæring af parametrene i CRF-modellen: stochastic gradient descent, contrastive divergence, pseudolikelihood og piecewise training.

Vores bidrag er tredelt: For det første formulerer og udleder vi den sande log-likelihood-objektfunktion og dens approksimationer i forhold til objektlokaliseringsproblemet. For det andet viser vi, at de approksimerende læringsmetoder opnår resultater, der er sammenlignelige med eksakt MCL læring, men kun contrastive divergence, pseudolikelihood og piecewise training er praktisk anvendelige. For det tredje viser vi, at vores resultater er sammenlignelige med resultaterne fra en structured support vector machine.

Foreword

This master's thesis marks the conclusion of our studies in the Copenhagen Master of Excellence Programme at Department of Computer Science, University of Copenhagen (DIKU). The actual work was carried out in the period February 23 to August 23 in close collaboration with the Lampert Group at Institute of Science and Technology Austria (IST Austria). IST Austria, inaugurated 2009, is an institute that focuses on basic research in the natural and mathematical sciences. The Lampert Group specializes in computer vision and machine learning research and this summer group leader Christoph Lampert was awarded the ERC Starting Grant.

We met Christoph Lampert in 2011 in Paris on a boat cruise organized by the ENS/INRIA Visual Recognition and Machine Learning Summer School where the idea for this collaborative project emerged. We presented the idea to our supervisor, Christian Igel, professor with special duties in machine learning at DIKU, who found it very interesting. The supervisor role has been split in two where Christian Igel has the main responsibility and Christoph Lampert manages the day-to-day supervision.

Since this is a joint master's thesis each of us has the main responsi-

bility of a specific part of the project. Andreas Christian Eilschou is mainly responsible for the log-likelihood approximations and Andreas Hjortgaard Danielsen is mainly responsible for the gradient approximations. This covers both implementation and presentation.

It goes without saying that we owe a lot of people thanks for helping bringing this thesis to life. First of all, a great many thanks goes to our supervisor Christoph Lampert for accepting us into his little army of researchers. We have enjoyed his personality and greatly appreciated his ability to, on the fly, perfectly explain any complicated topic. Also a great thanks goes to our supervisor Christian Igel for taking on the challenge of supervising a thesis across borders. Much information has been lost over the communication channel, and we know he will appreciate finally receiving this proof of our work.
With regards to other thesis related stuff, we thank Filip Korč, postdoc at IST Austria, for insightful discussions on probabilistic graphical models and Sebastian Nowozin, researcher at Microsoft Research, Cambridge, UK, for discussions on extensions of the CRF model and other applications. Also thanks to Kenny Erleben, associate professor at DIKU, for elaborations on LBFGS and line search methods.
Finally, we would like to thank everybody in the Lampert Group for introducing us to a life in research. You have made our stay at IST Austria a very pleasant experience and this experience might very well have changed our future careers! Last but not least, we absolutely appreciate the help we have received in developing our skills in after-lunch table football – six months of hard-core practice have really made a difference.

Andreas & Andreas
Klosterneuburg, Austria
August 23, 2012

Contents

1

Introduction

One of the principal goals of computer vision is to automatically interpret the contents of images on an abstract level. Detecting and localizing objects in the image is one of the key subtasks towards full image understanding. For example, the presence of multiple cars and bicycles gives a strong indication that the image depicts some kind of urban scene whereas the presence of cows and sheep indicates a country side scene. Besides detecting the presence of objects, finding their exact location in the image is important for determining interactions between objects, for instance, whether a person is riding a bicycle or just standing next to one. This thesis investigates a new model for the object localization problem using conditional random fields with focus on efficient parameter learning. In this chapter we give a brief overview of the thesis as well as an overview of related work for both object localization and parameter learning in conditional random fields.

1.1 Overview

Detecting objects in natural scenes is one of the key objectives in the annual PASCAL Visual Object Classes challenge [16]. In this setting,

(a) (b)

Figure 1.1 Natural images of a car in (a) and two cows in (b). Images are from the PASCAL VOC 2006 dataset.

detection is the combined task of deciding whether a specific object is present and if so, locate it by drawing a bounding box around it. Two examples of this are shown in Figure 1.1 where the task in the left image is two detect the red car and the task in the right image is to detect the two cows. Since we perform object localization in natural images we cannot assume anything about the number of objects or obtain any prior knowledge about the location of the objects. We are not even guaranteed that the entire object is visible. This makes the localization task especially challenging.

We propose a probabilistic view of the object localization problem in which we model the distribution of bounding boxes in a given image. Conditional random fields have long been popular for various computer vision tasks, including image denoising [33, 34], image segmentation [27, 43, 46] and object recognition [48, 50]. The CRF models in these problems are characterized by having many variables and thereby a large output space by which exact inference and exact maximum likelihood learning is often intractable in model. In the model we propose we have few variables, however, many states per variable and we thereby also face the problem of having a large output space and we too must look to approximate learning methods.

Luckily, the field of conditional random fields already gives us many different approximate learning techniques [33, 34, 35, 36, 43, 45, 53, 55,

61]. We look at two kinds of approximations: Approximations of the log-likelihood gradient where we optimize the real log-likelihood but use optimization algorithms that work on approximate gradients. Specifically, we look at stochastic gradient descent and contrastive divergence. Secondly, we look at approximating the log-likelihood by an alternative objective function that is tractable to optimize. In this category we investigate pseudolikelihood and piecewise training.

Our contributions of this thesis are the following: 1) we formulate the object localization problem in a probabilistic manner using conditional random fields, 2) we investigate popular approximate learning schemes for maximum conditional likelihood learning on two different datasets, and 3) we show that the parameters learned by contrastive divergence, pseudolikelihood and piecewise training give prediction performance similar to max-margin learning using structured support vector machines.

Throughout the text we will assume that the reader is familiar with basic multivariate statistics as well as basic machine learning methods for binary and multiclass classification. Furthermore, it will be helpful if the reader is somewhat familiar with probabilistic graphical models, especially Markov random fields. However, in Chapter 3 we will introduce Markov random fields and conditional random fields formally. We refer the reader to [5, 25] for general machine learning methods and to [32, 55, 62] for general probabilistic graphical models, Markov random fields and conditional random fields.

1.2 Related work

A lot of prior work exists on object localization and on approximate learning methods for conditional random fields. In the following we will only give a few examples.

1.2.1 Object localization

Rowley et al. [51] present a neural network for face detection. During training they use a bootstrapping technique to obtain negative examples by adding false detections to the training set. For testing they run the neural network classifier on a sliding window. Viola and Jones [60]

train a sliding window face detector using AdaBoost and introduce the integral image representation for rapid computation of region features.

Dalal and Triggs [13] introduce the histogram of oriented gradient (HOG) descriptors for use in human detection. The idea is to divide each possible region into a fixed number of histograms and use these histograms as features for a sliding window classifier, for example a binary support vector machine (SVM). Ferrari et al. [19] introduce a family of contour features for object detection with the motivation that some object categories are better discriminated by their shape than their interest points. They use these features to train a binary SVM.

Chum and Zisserman [11] propose an exemplar model that performs object localization in a weakly supervised manner that does not require annotations at training time. From images known to contain the object, the model learns similarities between the appearance of objects such that it can be used in an object detection system. They use the exemplar model to extract bounding boxes from images, represented by bag of visual words, to be used for training an SVM.

Since it is computationally inefficient to test every possible region with the sliding window, Lampert et al. [39] present a branch-and-bound strategy called the Efficient Subwindow Search (ESS). The strategy is to keep a priority queue of sets of regions, which is ordered by an upper bound on the quality of the region. Blaschko and Lampert [6] use the ESS to efficiently train a structured SVM for object localization.

1.2.2 Conditional random fields in computer vision

Kumar and Hebert [34, 35] introduce CRFs to computer vision with their Discriminative Random Field (DRF) model used for image denoising and detection of man-made structures. The DRF model relaxes the conditional independency assumptions usually inherent in MRF models and this allows the model to make better use of the data. Parameter learning is performed by the maximum pseudolikelihood method. He et al. [27] use a multiscale CRF with hidden variables for image segmentation and used contrastive divergence for parameter learning.

Quattoni et al. [48] use CRFs for recognition of a single object in an image. They propose a part-based CRF and introduce hidden

variables to model the parts which are not observed at training time. This makes the log-likelihood non-convex and therefore maximizing the log-likelihood is not guaranteed to find the global optimum. Another application is by Wang and Gong [63] who use hidden state CRFs for natural scene categorization. Their proposed model classifies the entire input with a single label, compared to previous approaches that label segments of the input, for example image patches. For training they use a quasi-Newton method and they use belief propagation to compute marginals and the log-partition function.

A CRF model for detecting objects in multi-camera scenarios is proposed by Roig et al. [50]. The parameters are estimated by max-margin learning. Also, Huang et al. [30] use a hierarchical CRF for segmenting and labeling street scenes and show an application on computing image similarity. They compute a labeling on the images using one CRF and then cluster the images based on this labeling and re-label the images using another CRF trained on the individual clusters.

1.2.3 CRF learning

None of the papers above discuss learning in detail and they do not compare different learning algorithms. Parise and Welling [45] were the first (to our knowledge) to investigate and compare different learning methods for MRFs. They create two models, one fully observed and one with hidden variables, both of which they can create perfect samples from. They compare pseudolikelihood (PL), contrastive divergence (CD) and pseudo-moment matching (PMM) against the exact log-likelihood. They conclude that the performance of PL and CD is comparable with exact maximum likelihood for the fully observed models and that CD is preferable when using hidden variables.

Kumar et al. [34] review gradient approximations using inference techniques for the DRF model. They show that the performance of the learning algorithm is highly dependent on the inference method used for computing the expectation in the gradient. A similar investigation was performed by Korč and Förstner [33] where they show that penalized pseudolikelihood outperforms the inference-based methods proposed by Kumar et al. [34].

Stochastic gradient methods for CRF training has been suggested by Bottou and Bousquet [9] and Vishwanathan [61] and has been shown to accelerate the learning process for CRFs when the exact inference is possible. When this is not the case, careful selection of the learning rate is needed.

Sutton and McCallum [54] introduce piecewise training as a transformation of the probabilistic graphical model into a new model consisting of piecewise factors called the node-split graph in which true maximum likelihood learning is performed. The parameters obtained are an approximate solution to the real maximum log-likelihood. Sutton and McCallum [53] also introduce piecewise pseudolikelihood in which maximum pseudolikelihood is performed in the node-split graph.

Finally, Nowozin et al. [43] compare parameter learning methods for the problem of object class image segmentation using a hierarchical CRF. They show that piecewise training gives results similar to exact maximum likelihood but with a reduced training time. Furthermore, they compare with max-margin learning and show that CRF learning outperforms max-margin learning for problems with many parameters.

1.3 Outline

This thesis is structured as follows: In Chapter 2 we explain image representation including the bag of visual words, we introduce the object localization problem as a maximization problem, as well as how to one evaluates the quality of a bounding box. In Chapter 3 we introduce the conditional random field model of the bounding box distribution and discuss maximum conditional likelihood learning. We examine the computational complexity and discuss why maximum conditional likelihood is in general intractable. Gradient approximations of the learning problem are presented in Chapter 4 and we show how to approximately optimize the conditional log-likelihood using stochastic gradient descent and contrastive divergence. In Chapter 5 we show how maximum pseudolikelihood and piecewise training approximately solve the maximum conditional likelihood problem. Finally, we show and discuss experimental results on two object localization datasets in Chapter 6 and Chapter 7.

2

Object localization

In this chapter we describe the problem of object localization in the context of computer vision and machine learning. We show how we represent images and how a model can be trained in order to predict bounding boxes on these images. We also show how we measure the performance of these localizations.

2.1 Introduction

Given a natural image in which an instance of a particular object category is present, object localization is the problem of determining the region of the image containing this object. In Figure 2.1 we see an image of a street scene with an of instance of `person`, `bus`, and `car`, each of which we want to locate.

The location of the object in the image can be represented in numerous ways: by its center point, its contour, a bounding box, or by a pixel-wise segmentation. We use the bounding box representation that bounds a rectangular region of the image by the four coordinates that define the position of the left, top, right, and bottom side of the box. This representation may be simple, however, it is a widely used ap-

Figure 2.1 Object categories `person`, `bus` and `car` in an image from the PASCAL VOC 2006 dataset.

proach and the dependencies between the sides of the bounding box provides us with a structure that will prove advantageous. See Figure 2.2 for examples of bounding boxes in the street scene image for each of the three object instances.

In machine learning we address object localization as a supervised learning problem with two stages: a learning stage and a prediction stage. In the learning stage we are given a training set of images with annotations of the location of a given object. From this training set the goal is to learn a prediction function that predicts the most likely bounding box on a previously unseen image containing the object. To formalize:

- We are given N training images $x^1, \ldots, x^N$ with each $x^n \in \mathcal{X}$.
- For each training image x^n we have the annotation y^n that is the location of the object category in the image.
- Each $y^n \in \mathcal{Y} = \mathbb{N}^4$ specifies a bounding box by the *left, top,*

(a) `person` (b) `bus` (c) `car`

Figure 2.2 The object categories `person`, `bus` and `car` located with bounding boxes.

right, bottom coordinates.

- We learn a prediction function $f : \mathcal{X} \to \mathcal{Y}$ that takes a test image $x \in \mathcal{X}$ as input and predicts the most likely bounding box $y \in \mathcal{Y}$.

At training time we assume one object instance per image and if more instances are present we treat each instance separately.

The problem of object localization is closely related to the problem of object detection. In object detection we want to determine whether a member of a particular object category is present in an image or not, and if it is, we want to locate it as well. In other words, object detection is a problem consisting of a binary object classification step followed by a localization step.

2.2 Image representations

For object localization in natural images we are faced with the difficult task of generalizing the appearance of a category over many different images. We need to represent the images in a convenient way. Using the intensities of the image pixels directly as features is not a sensible approach for multiple reasons. One problem is that the images come in different resolutions. Another problem is that we are presented with a

very high dimensional feature space; large images would be computationally hard to work with and we would also experience the *curse of dimensionality* by not having enough training data to be able to cover the space. Thereby, we would not be able to learn to generalize how an object category is to be predicted. We need to extract a smaller number of features from the images and these features need to be discriminative for the object category.

For the same object occurring in multiple natural images there will be variations in scale, rotation, projection, imaging, noise level, illumination and occlusion depending on how each image was generated. Additionally, there is the intra-category variation. For instance, as seen in Figure 2.3 a fluffy little cavachon and a big boxer do not seem to look that alike, however, they both belong to the object category `dog`. Localization should be able to handle all these variations and learn the similarities within a certain object category and therefore we need to extract features that facilitate this discrimination.

We represent an image x by a collection of M local feature points $\{p_1, \ldots, p_M\}$ where each $p_m = (l_m, c_m)$ consists of a location l_m (the pixel coordinates in the image grid) and a quantity c_m expressing the appearance of the region. In the bag of visual words [12] representation, c_m corresponds to a reference into an appearance codebook such that $c_m = k$ if the image region at l_m looks like the k'th element of the codebook. We use this bag of visual words representation in order to compare our results with those of Blaschko and Lampert [6].

In spite of its simplicity, the bag of visual words representation has shown good results for object classification [1, 15, 18, 52, 66] even though some methods are known to be better at detecting specific categories [11, 13, 19].

2.2.1 Bag of visual words

The paper by Csurka et al. [12] presents a bag of keypoints approach to generic visual categorization. The bag of keypoints is also known as a bag of visual words to emphasize the relation to the original use of bag of words in text categorization. We will adopt this bag of visual words representation to the problem of object localization with the main steps

(a) cavachon (b) boxer

(c) pug (d) sheep

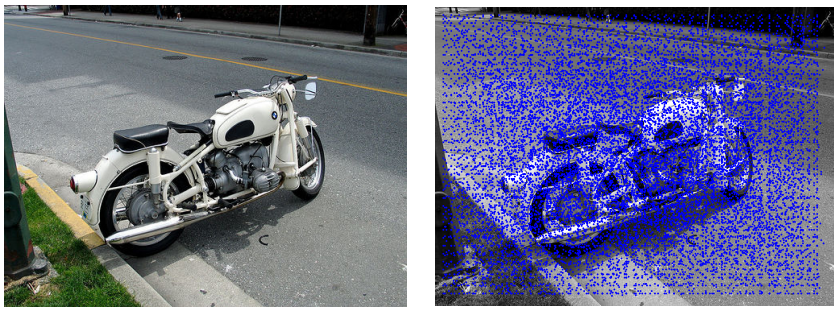
Figure 2.3 One of the challenges of object localization in natural images is the intra-class variation. The cavachon in (a) looks very different from the boxer in (b) and the pug in (c), however, they all belong to the object category `dog`. At the same time, the cavachon shares a resemblance resemblance to the `sheep` in (d). The images are from the PASCAL VOC 2006 dataset.

being:

- Detection of image interest points.
- Description of the interest points.
- Assignment of the descriptors to a vocabulary of visual words.
- Construction of the bag of visual words which counts the number of descriptors assigned to each visual word.

Ideally, the interest points should be repeatable in the sense that if there is a transformation between two images of an object, corresponding interest points are detected in both images. The extracted descriptors should be invariant to the variations that are irrelevant to the localization task (3D camera viewpoint, illumination, occlusion, etc.) while at the same time being distinctive enough to enable us to locate the object in the image.

The vocabulary is created by performing K-means [5] clustering on the descriptors of the training set, where the number clusters K is to be decided upon. The cluster centers will then be the visual words of the vocabulary. This vocabulary should be large enough to distinguish relevant changes in descriptors while not being too large such that we capture noise. Given a previously unseen image, we compute the descriptors in a given region of the image, assign each descriptor to the nearest visual word, and compute the bag of visual words as a histogram of the visual words. This gives us the advantage for the object localization task that even if we do not have an equal number of descriptors within each image region, we will always have a feature vector that has a fixed size, namely the size of the vocabulary.



(a) original (b) interest points

Figure 2.4 An image of an object `motorbike` in (a) along with its interest points in (b) for which we will find visual words. Original image is from the PASCAL VOC 2006 dataset.

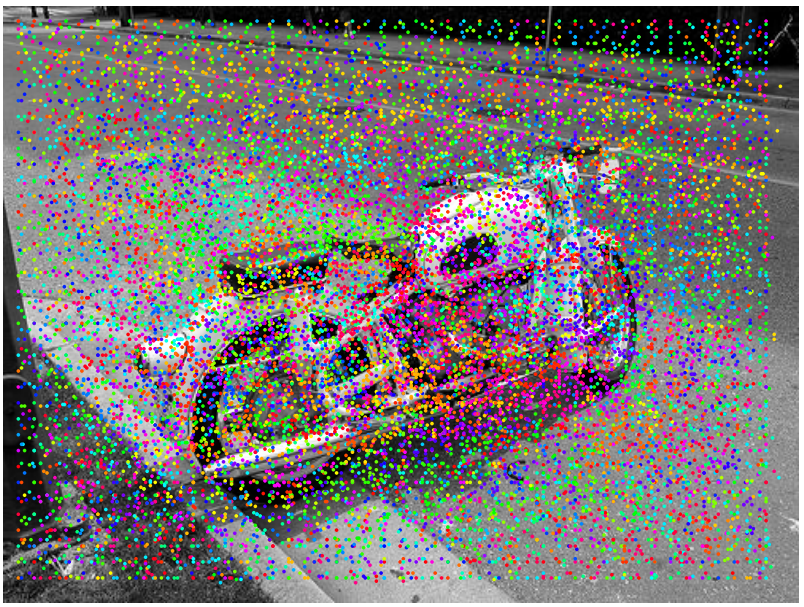


Figure 2.5 An image of an object `motorbike` along with the visual words that each interest point has been assigned to based on the local descriptors. Each color level represents one of the 3,000 visual words.

2.2.2 Dataset specifics

The structured support vector machine used by Blaschko and Lampert [6] was tested on the PASCAL VOC 2006 and the TU Darmstadt `cow` dataset where they obtained the descriptors using the fast method of Upright Speeded Up Robust Features (U-SURF) [1], which is a scale-invariant only descriptor. In these datasets rotational invariance is not required since the categories (`bus`, `car`, `bicycle`, `cow`, etc) are most likely to be found upright in the images.

Blaschko and Lampert [6] extracted the U-SURF descriptors both at the interest points found by U-SURF and also on a regular grid and at random locations. They sampled 100,000 descriptors from training images and clustered them using K-means clustering into a vocabulary of 3,000 visual words. See Figure 2.4 for an example of the interest points chosen for an image and see Figure 2.5 for an illustration of

which visual words these interest points have been assigned to.

2.3 Prediction

Assume that we represent an image by the bag of visual words representation and that for each visual word k we have a weight $w_k \in \mathbb{R}$, where $w_k > 0$ means *looks like the object* and $w_k < 0$ means *does not look like the object*. We can form a prediction function $f : \mathcal{X} \to \mathcal{Y}$ by first defining a quality function $g : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ by

$$g(x, y) = \sum_{m=1}^{M} w_{c_m} \mathbb{1}_{[l_m \in y]}, \tag{2.1}$$

where $\mathbb{1}_{[l_m \in y]}$ is 1 if the visual word c_m located at l_m is within the region y and 0 if it is not. That is, taking an image x and a region y within that image, the function $g(x, y)$ corresponds to summing up the weights of the visual words that fall into the region of y. Since we assume that w_k reflects how much a visual word looks like it comes from the particular object category, the value of $g(x, y)$ will be high if the region y contains many points that look like they belong to the object category and few points that look like they do not belong to the object category.

Let us introduce a feature map $\varphi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^D$ that builds the histogram of visual words in the region $y \in \mathcal{Y}$ in the image $x \in \mathcal{X}$. Then we can express the right-hand side of Equation (2.1) as a dot product between the weights and this bag of visual words

$$g(x, y) = \langle w, \varphi(x, y) \rangle. \tag{2.2}$$

We can now define the prediction function $f(x)$ using the quality function $g(x, y)$

$$f(x) = \arg\max_{y \in \mathcal{Y}} g(x, y), \tag{2.3}$$

such that we in a new image identify the region with the highest score and predict this region as the object location. We will use the branch-and-bound strategy of Efficient Subwindow Search [39] to locate this region.

2.3.1 Integral image

During both prediction and learning we will have to evaluate the quality function g from Equation (2.3) multiple times and each time we will have to compute the dot product of the weights and the bag of visual words within a certain region. The computation of the bag of visual words requires that we run through the feature points and increment the bin of a visual word whenever we have one located within the region. A more efficient approach to computing these dot products is to construct an intermediate image representation, the *integral image*, as presented by Viola and Jones [60]. If we return to the definition of g in Equation (2.1), the integral image $\mathcal{I}$ at location (i, j) contains the sum of the weights of visual words located above and left of (i, j) inclusive:

$$\mathcal{I}(i,j) = \sum_{\substack{i' \leq i \\ j' \leq j}} w_{c_{i',j'}}, \tag{2.4}$$

where we assume the weight $w_{c_{i,j}}$ to be zero if no visual word is located at (i, j).

Once we have this integral image, the dot product of the weights with the bag of visual words within any region y defined by the left-top corner (i_L, j_T) and the right-bottom corner (i_R, j_B) can be found by just four lookups, namely

$$\langle w, \varphi(x,y) \rangle = \mathcal{I}(i_L, j_T) - \mathcal{I}(i_L, j_B) - \mathcal{I}(i_R, j_T) + \mathcal{I}(i_R, j_B) \tag{2.5}$$

As an example see Figure 2.6 where $\mathcal{I}(i_L, j_T)$ has region A, $\mathcal{I}(i_L, j_B)$ has region A+C, $\mathcal{I}(i_R, j_T)$ has region A+B, and $\mathcal{I}(i_R, j_B)$ has region A+B+C+D. By substituting into Equation (2.5) we obtain the region D as desired.

2.4 Learning

In order to predict good bounding boxes on previously unseen images, we need to learn the weight vector w from the training set. So far, two learning methods are being used for this: *binary classifier training* and *structured output learning*.

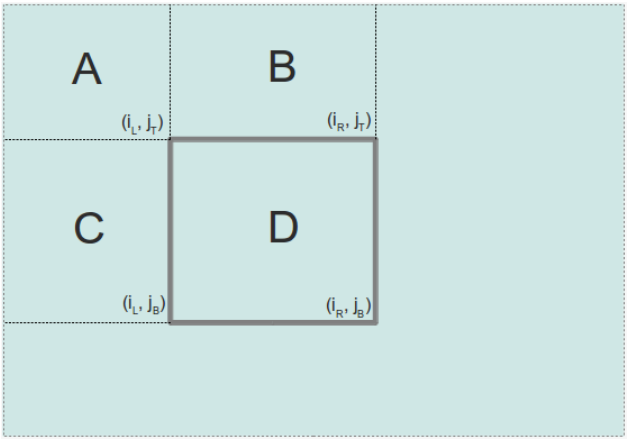


Figure 2.6 Integral image. The sum of the weights of visual words within any region of the image can be computed by four lookups.

2.4.1 Binary classifier training

Binary classifier training is a widely used method for object detection [8, 11, 13, 19, 51, 60]. Learning is achieved by training for instance a support vector machine on a number of positive and negative regions of the images. The weights are learned such that the quality function gives $g(x, y) > 1$ for all object regions in the training set and $g(x, y) < -1$ for all regions where the object is not present.

A main drawback of this approach is that it is unclear what to do with non-object regions that partially overlap the object region. The positive training examples are easy to obtain; they are simply the annotations in the training set. The problem is the negative examples, which are not given. It is difficult to know which negative training examples to choose. Often bootstrapping is performed where we train and test multiple times while adding the misclassifications to the training set [51]. As a consequence of this, obtaining a good training set is cumbersome.

2.4.2 Structured output learning

Structured output learning using the method of Blaschko and Lampert [6] of the structured support vector machine overcomes the problems of binary classifier training by taking into account all possible image regions. The structured support vector machine learns weights such that $g(x, y) \geq g(x, y') + \Delta(y, y')$ whenever y is an object region in the image x, and y' is a non-object region, and $\Delta(y, y')$ measures how much y and y' overlap.

Since all bounding boxes are considered while learning, the problem of regions overlapping non-object and object regions is not an issue anymore, however, the approach is computationally expensive. Also, the method is designed to detect only one object per image and it lacks a probabilistic interpretation. By modeling the distribution of bounding boxes as a conditional random field, we can perform structured output learning with a probabilistic interpretation as we will see in the following chapters. See Appendix A for a more thorough description of the structured support vector machine.

2.5 Performance evaluation

To evaluate the quality of the localization we use the official PASCAL measure for object detection, namely the area overlap defined as

$$\text{overlap}(y^n, y) = \frac{\text{Area}(y^n \cap y)}{\text{Area}(y^n \cup y)} \tag{2.6}$$

where y^n is the ground truth bounding box in image x^n and y is the predicted bounding box. Since we do not perform detection but only localization, we will measure the total performance on the dataset by the overlap-precision measure inspired by Vedaldi et al. [59]. The overlap-precision measure is computed as follows: for any overlap $\theta \in [0, 1]$ we measure the fraction of boxes with minimum θ overlap with a ground truth box. We assume one box per image, so if more objects are present in the image we choose the ground truth as the box that gives the largest overlap. The precision is the number of boxes with overlap larger than θ divided by the total number of predictions, which in our setup is one per image. In Figure 6.3 we see an example of an overlap-precision

curve. To get one score for evaluating performance on the test set we can compute the area under the curve (AUC). This area will have a value between zero and one and will be exactly one for perfect prediction performance.

2.6 Summary

We introduced the problem of object localization in natural images, which is a difficult task with both intra-class variations and great variations in the image acquisition. An image representation that allows discriminating between object regions and non-object regions is needed and to compare with previous methods we choose U-SURF features. We presented the prediction task as the maximization of a quality function and discussed parameter learning using a binary classifier and a structured support vector machine. Finally, we introduced the bounding box overlap to measure prediction performance.

3

Conditional random fields

In this chapter we introduce the concept of structured output learning based on probabilistic models. Specifically, we introduce the framework of conditional random fields and show how we can model the bounding box distribution in an image. We then discuss maximum conditional likelihood learning and show that it is often intractable to solve this optimization problem exactly.

3.1 Introduction

Prediction has traditionally been divided into two categories: Regression, where you predict a continuous value and classification where you predict a value from a discrete set of classes. That is, you predict one single value or a number of independent values. As discussed in the previous chapter, predicting a bounding box requires determining four coordinates, namely the top-left corner and the bottom-right corner of the bounding box. That is, four numbers that are highly dependent on each other.

As we saw in Chapter 2 we need to estimate the prediction function $f : \mathcal{X} \to \mathcal{Y}$ in Equation (2.3) where the output domain $\mathcal{Y}$ is more than

simply a class variable. In this case we call $\mathcal{Y}$ a structured output domain and the prediction problem is known as *structured prediction* [58]. We focus on the class of structured prediction methods that model the problem in a probabilistic framework called a conditional random field.

The quality function $g(x, y)$ is parameterized by a weight vector w that needs to be learned from training data. In the probabilistic setting we can use maximum conditional likelihood learning to estimate w. For a thorough treatment of structured prediction and learning in computer vision, see the tutorial by Nowozin and Lampert [44].

3.2 Probabilistic graphical models

Probabilistic graphical models present a powerful framework for modeling the independencies between random variables in a probability distribution using graph theory. The idea is to let each random variable be represented by a node in the graph and to let edges between nodes represent dependencies between the random variables. Graphical models are used in a large variety of fields besides computer vision, for example natural language processing, speech processing and bioinformatics and there exist several textbooks on general graphical models, see for example [5, 32, 62].

To set up a graphical model for the prediction problem we assume that we have two sets of random variables as discussed above: A set of input random variables $X = \{X_1, X_2, ..., X_n\}$ taking values in the input domain $\mathcal{X}$ and a set of output random variables $Y = \{Y_1, Y_2, ..., Y_m\}$ taking values in the output domain $\mathcal{Y}$. Collectively, we denote these variables as $V = X \cup Y$ and let $E = V \times V$ denote the edges between these variables. Then we can construct a graph $G = (V, E)$ that represents a family of probability distributions by the random variables and the dependencies (edges) between them. In the following we use the shorthand $p(v)$ to denote $p(V = v)$ and $p(v_i)$ to denote $p(V_i = v_i)$ and similarly for $p(x, y)$. With slight abuse of notation we will also use $i \in V$ to denote the indices of the variables in V.

The edges can both be directed and undirected but we will concentrate on the undirected models, often called *Markov random fields* (MRFs) or Markov networks. We will then look at a discriminative

variant of MRFs called conditional random fields.

3.2.1 Markov random fields

Let $G = (V, E)$ be an undirected graph and let

$$N(i) = \{j \in V \mid (i, j) \in E\} \tag{3.1}$$

denote the neighborhood of V_i. Then G is called a *Markov random field* if it satisfies the following conditional independence property:

$$p(v_i|v_{V\setminus i}) = p(v_i|v_{N(i)}) \tag{3.2}$$

where $v_{V\setminus i} = (v_1, v_2, \ldots, v_{i-1}, v_{i+1}, \ldots)$.

We define the notion of a clique c in graph G as a subset of V such that there is an edge between every pair of variables in the clique. With this definition we define $C(G)$ to be the set of maximal cliques in G. The classical Hammersley-Clifford theorem states that the MRF defined as above defines the family of probability distributions that factorize as follows [23, 55]:

$$p(v) = \frac{1}{Z} \prod_{c \in C(G)} \Psi_c(v_c) \tag{3.3}$$

where $\Psi_c(v_c)$ is called the *potential function* over the clique c and

$$Z = \sum_{v \in \mathcal{V}} \prod_{c \in C(G)} \Psi_c(v_c) \tag{3.4}$$

is a normalization constant called the *partition function* where the sum is over all possible values of V, that is $\mathcal{V} = \mathcal{X} \cup \mathcal{Y}$. This family of distributions is called *Gibbs distributions*.

The presence of edges represents a dependency. In Figure 3.1 we see two examples of graphical models with four nodes. The left graph is a fully connected graph which is able to represent all distributions of the form Equation (3.3) with four random variables. In the graph on the right we see that there is no edge between V_1 and V_4 and between V_2 and V_3. But the variables are still dependent on each other if we do not know anything about their values. If, on the other hand, we observe the values of V_2 and V_3, then V_1 and V_4 are still dependent in

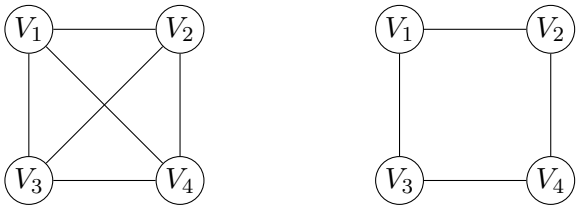


Figure 3.1 A fully connected graph (left) and a graph with pairwise dependencies between some vertices (right). The vertices in the graph correspond to random variables and the edges correspond to dependencies between the random variables.

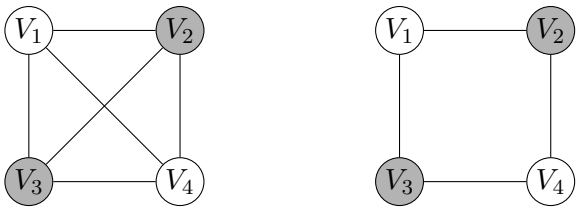


Figure 3.2 A fully connected graph (left) and a graph with pairwise dependencies between some vertices (right) where V_2 and V_3 have been observed. Observed variables are colored a light gray.

the fully connected graph whereas they become independent of each other in the other graph. This is shown in Figure 3.2 and this notion of conditional independence is an important property that we will utilize in the pseudolikelihood approximation in Chapter 5.

Since we model both the input and the output variables, collectively denoted V, the Markov random field as described so far models $p(x, y) = p(y|x)p(x)$ which means that we implicitly model the distribution of the input variables as well as the output variables. We call such a model a *generative* model since we can generate all kinds of instances (both input and output) given the distribution. This is not always advantageous since the distribution of the input variables may very well be quite complicated. Modeling the distribution of natural images, for example, is non-trivial and usually you would have to impose some independence assumptions [55]. Before discussing the conditional random fields, which avoid modeling $p(x)$ explicitly, we will introduce the factor graph representation which makes the factorization explicit.

3.2.2 Factor graphs

We extend the graph structure with an additional set of nodes $\mathcal{F}$ called *factors* such that each variable V_i is connected to a factor $F \in \mathcal{F}$. The resulting bipartite graph $G = (V, \mathcal{F}, E)$ is called a *factor graph*. To each factor $F \in \mathcal{F}$ we define a *potential function* $\Psi_F(v_{N(F)})$ where

$$N(F) = \{i \in V \mid (i, F) \in E\} \tag{3.5}$$

is called the *scope* of the factor F.

Given a factor graph G we can now define the family of probability distributions it represents by the factorization

$$p(v) = \frac{1}{Z} \prod_{F \in \mathcal{F}} \Psi_F(v_{N(F)}) \tag{3.6}$$

where the partition function is given by

$$Z = \sum_{v \in \mathcal{V}} \prod_{F \in \mathcal{F}} \Psi_F(v_{N(F)}). \tag{3.7}$$

In Figure 3.3 we see two examples of factorizations of a fully connected graph. From these two graphs we can directly write down the distributions. For the graph on the left we have just a single factor, and therefore the distribution is given by

$$p(v) = \frac{1}{Z} \Psi(v_1, v_2, v_3, v_4) \tag{3.8}$$

For the graph on the right we have six factors and thus the distribution becomes

$$\begin{aligned} p(v) = \frac{1}{Z} &\Psi_{1,2}(v_1, v_2) \Psi_{1,3}(v_1, v_3) \Psi_{1,4}(v_1, v_4) \\ &\cdot \Psi_{2,3}(v_2, v_3) \Psi_{2,4}(v_2, v_4) \Psi_{3,4}(v_3, v_4) \end{aligned} \tag{3.9}$$

The factor graph gives us a direct correspondence between the factors in the graph and the potential functions, whereas the normal graph representation used in Figure 3.1 does not.

3.2.3 Conditional random fields

Since the input images are always known at training and test time we do not need to model the marginal probability distribution of the

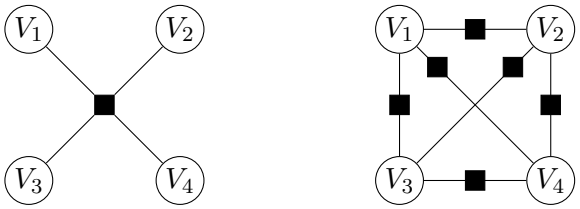


Figure 3.3 An example of a fully connected graph represented by one factor (left) and six factors (right).

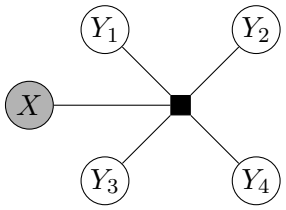


Figure 3.4 CRF represented as a factor graph.

images $p(x)$ for the purpose of prediction. However, we are interested in modeling the distribution of the bounding boxes Y given a specific image, $p(y|x)$. This gives rise to the *conditional random field* model which models Y conditioned on X [37, 55] as

$$p(y|x) = \frac{1}{Z(x)} \prod_{F \in \mathcal{F}} \Psi_F(x_{N(F)}, y_{N(F)}) \tag{3.10}$$

and

$$Z(x) = \sum_{y \in \mathcal{Y}} \prod_{F \in \mathcal{F}} \Psi_F(x_{N(F)}, y_{N(F)}) \tag{3.11}$$

Thus, CRFs can be seen as a discriminative variant of MRFs where we condition on the inputs. We see that the only thing that changes is the partition function which is now a function of the input x and needs to be recomputed for each image. An example of a CRF modeled as a factor graph is shown in Figure 3.4.

When defining the probability distribution for a given problem, one needs to decide on two things: the structure of the graphical model and the form of the potentials. The graphical structure may be easily decided from the problem at hand but the form of the potentials, that is,

the specific dependencies between the random variables may not be as easily modeled. A common form of the potentials is $\Psi_F(x_{N(F)}, y_{N(F)}) = \exp(g_F(x, y))$ which results in the following CRF model:

$$p(y|x) = \frac{1}{Z(x)} \prod_{F \in \mathcal{F}} \exp(g_F(x_{N(F)}, y_{N(F)})) \tag{3.12}$$

$$= \frac{1}{Z(x)} \exp\left(\sum_{F \in \mathcal{F}} g_F(x_{N(F)}, y_{N(F)}) \right) \tag{3.13}$$

Note that often the function $g_F(x, y)$ is defined as $-E(x, y)$ where $E(x, y)$ is an energy function that we want to minimize.

3.2.4 Object localization

We can now set up our object localization model. As we saw in the previous chapter, we have a quality function $g(x, y) = \langle w, \varphi(x, y) \rangle$ that determines how well the bounding box y captures the desired object in image x. Here w is a D-dimensional vector and $\varphi(x, y)$ is a bag of visual words representation of the bounding box y in image x. We can interpret this as a single factor in our bounding box distribution and substituting into (3.13) we obtain the following distribution of bounding boxes:

$$p(y|x, w) = \frac{1}{Z(x, w)} \exp\left(g(x, y)\right) \tag{3.14}$$

$$= \frac{1}{Z(x, w)} \exp\left(\langle w, \varphi(x, y) \rangle\right) \tag{3.15}$$

Note that we have also conditioned on the weight vector w which now parameterizes the distribution. In Figure 3.5 we see the distribution represented as a factor graph conditioned on the input image x. Since the weight vector is implicit in the factor we do not model it in the graphical model. However, this would make sense to do in a Bayesian setting where we would impose a prior distribution over w. We will return to this notion in Section 3.4.

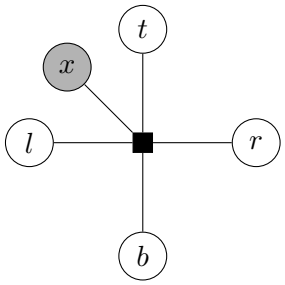


Figure 3.5 Probabilistic graphical model of the bounding box.

3.3 Gibbs sampling

From the distribution of bounding boxes we can draw samples which gives us the ability to investigate how bounding boxes are placed in an image for a given weight configuration. For example, we can investigate whether an image with two object instances will have two modes centered on these objects and in that case how sharp the modes are. Direct sampling from the model distribution is not feasible and therefore we must resort to *Markov Chain Monte Carlo* (MCMC) methods.

The main idea behind MCMC methods is to construct a Markov chain of samples $\tilde{y}^{(k)}$ that has been sampled from some distribution $q(y|\tilde{y}^{(k-1)})$ starting at some random configuration. The distribution $q(y|\tilde{y}^{(k-1)})$ must be chosen such that we achieve a Markov chain that has as its stationary distribution $p(y|x, w)$. Thus, to get a sample from the model distribution we need to take a large amount of steps. It is therefore important that each sampling step is not too time consuming. Here we will discuss an efficient MCMC sampler called the *Gibbs sampler* and show how it can be used to sample bounding boxes in an image.

Originally invented by Geman and Geman [23] in 1984 for image restoration, the Gibbs sampler is a special case of the Metropolis-Hastings sampler [26] in which samples are drawn from a proposal distribution and accepted with a certain probability. The Gibbs sampler chooses this proposal distribution to be the conditional distribution of one variable given the rest and this causes each sample to be accepted with probability 1.

Let $p(y|x,w) = p(y_1, y_2, \ldots, y_M|x,w)$ be the model distribution over M variables. The Gibbs sampler then samples from the one-dimensional distribution $q(y|\tilde{y}^{(k-1)}) = p(y_s|\tilde{y}_{\neg s}^{(k-1)}, x, w)$ where $\tilde{y}_{\neg s}^{(k-1)}$ denotes all variables of the $k-1$'st sample except the s'th variable. Sampling all variables from this Gibbs chain therefore results in the following samples:

$$\begin{aligned}
\tilde{y}_1^{(k)} &\sim p(y_1|\tilde{y}_2^{(k-1)}, \ldots, \tilde{y}_M^{(k-1)}, x, w) \\
\tilde{y}_2^{(k)} &\sim p(y_2|\tilde{y}_1^{(k)}, \tilde{y}_3^{(k-1)} \ldots, \tilde{y}_M^{(k-1)}, x, w) \\
\tilde{y}_3^{(k)} &\sim p(y_3|\tilde{y}_1^{(k)}, \tilde{y}_2^{(k)}, \tilde{y}_4^{(k-1)}, \ldots, \tilde{y}_M^{(k-1)}, x, w) \\
&\vdots \\
\tilde{y}_M^{(k)} &\sim p(y_d|\tilde{y}_1^{(k)}, \tilde{y}_2^{(k)} \ldots, \tilde{y}_{M-1}^{(k)}, x, w)
\end{aligned} \tag{3.16}$$

This series of intermediate samples is called a *sweep* of the Gibbs sampler.

The question now is how to sample from the one-dimensional distribution $p(y_s|\tilde{y}_{\neg s}^{(k-1)}, x, w)$. We perform *inversion by sequential search* [14] which is a method applicable for generating samples from any discrete one-dimensional distribution. The algorithm looks like this:

(1) Calculate the cumulative distribution

$$F(i) = p(y_s \leq i|\tilde{y}_{\neg s}^{(k-1)}, x, w) \tag{3.17}$$

for all values that y_s can take.
(2) Generate a random number u from the uniform distribution in the interval $[0, 1]$.
(3) Initialize i at the smallest value of y_s.
(4) Increment i until $u \leq F(i)$.
(5) Set $\tilde{y}_s^{(k)} = i$.

We see that the complexity of this sampling scheme is linear in the number of values y_s can take.

In Figure 3.6 we see two examples of samples obtained from the bounding box distribution. Each sample was obtained by running the Gibbs chain from a random configuration for 1000 steps. We will return to this in our discussions in Chapter 7.

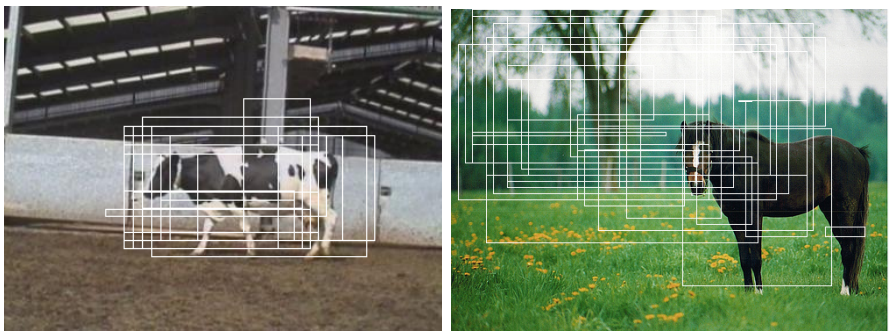


Figure 3.6 Samples from the bounding box distribution. The distribution for the cow obtained from the TU Darmstadt dataset seems to be quite peaked around the cow. But the horse distribution obtained from the PASCAL VOC 2006 dataset seems to be covering the tree instead of the horse.

As we will see in the next section on parameter learning, the derivative of the log partition function $\log Z(x, w)$ takes the form of an expectation over the model distribution and therefore sums over all possible output configurations. Because of the large output space this easily becomes intractable to compute directly. One way to alleviate this problem is to approximate the expectation by the mean over S example outputs sampled from the model distribution $p(y|x, w)$. The contrastive divergence method makes use of the Gibbs sampler to approximate this expectation.

3.4 Maximum conditional likelihood learning

Until now we have assumed that the model distribution $p(y|x, w)$ correctly models the distribution of bounding boxes in a given image. We will now show how to use maximum conditional likelihood learning to estimate the parameters w of this distribution. Assume that we are given a dataset $\mathcal{D} = \{(x^n, y^n)|n = 1, 2, ..., N\}$ where each data pair is independent and identically distributed according to the unknown distribution $d(x, y)$ called the *data distribution*. We want to find the parameters w that makes $p(y|x, w)$ as close to $d(y|x)$ as possible. To do this we use the Kullback-Leibler (KL) divergence to measure dissimilarity between distributions, which for a given input $x \in \mathcal{X}$ is given

as

$$\mathrm{KL}(p||d) = \sum_{y \in \mathcal{Y}} d(y|x) \log \frac{d(y|x)}{p(y|x,w)}. \tag{3.18}$$

The expected dissimilarity over all inputs is given as

$$\begin{aligned}
\mathrm{KL}_{\mathrm{total}}(p||d) &= \sum_{x \in \mathcal{X}} d(x) \sum_{y \in \mathcal{Y}} d(y|x) \log \frac{d(y|x)}{p(y|x,w)} && (3.19)\\
&= \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} d(x,y) \log d(y|x) \\
&\quad - \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} d(x,y) \log p(y|x,w) && (3.20)\\
&= \mathbb{E}_{(x,y)\sim d(x,y)} \left[\log d(y|x)\right] - \mathbb{E}_{(x,y)\sim d(x,y)} \left[\log p(y|x,w)\right] && (3.21)
\end{aligned}$$

and measures the total dissimilarity between the distributions. Since the first term does not depend on w we can ignore it and with the assumption, that the data points are sampled from $d(x,y)$, the second term can be approximated by the empirical mean using these N samples. Thus, to find the set of parameters w^* that minimizes the dissimilarity between d and p we have the following optimization problem:

$$\begin{aligned}
w^* &= \arg\min_{w \in \mathbb{R}^D} \mathrm{KL}_{\mathrm{total}}(p||d) && (3.22)\\
&= \arg\max_{w \in \mathbb{R}^D} \sum_{n=1}^{N} \log p(y^n|x^n,w) && (3.23)\\
&= \arg\max_{w \in \mathbb{R}^D} \prod_{n=1}^{N} p(y^n|x^n,w) && (3.24)
\end{aligned}$$

which we call the *maximum conditional likelihood* (MCL). We can pose the optimization problem as a minimization problem by minimizing the

negative conditional log-likelihood $\mathcal{L}(w)$:

$$\mathcal{L}(w) = \sum_{n=1}^{N} -\log p(y^n|x^n, w) \tag{3.25}$$

$$= \sum_{n=1}^{N} -\log \left(\frac{1}{Z(x^n, w)} \exp(\langle w, \varphi(x^n, y^n)\rangle) \right) \tag{3.26}$$

$$= \sum_{n=1}^{N} -\langle w, \varphi(x^n, y^n)\rangle + \sum_{n=1}^{N} \log Z(x^n, w). \tag{3.27}$$

It can be shown that $\mathcal{L}(w)$ is a convex function [44] and therefore we can use gradient descent methods to solve the optimization problem. We need to calculate the gradient of $\mathcal{L}(w)$ with respect to the weights:

$$\nabla_w \mathcal{L}(w) = \frac{\partial}{\partial w} \left(-\sum_{n=1}^{N} \langle w, \varphi(x^n, y^n)\rangle + \sum_{n=1}^{N} \log Z(x^n, w) \right) \tag{3.28}$$

$$= -\sum_{n=1}^{N} \frac{\partial}{\partial w} \langle w, \varphi(x^n, y^n)\rangle + \sum_{n=1}^{N} \frac{\partial}{\partial w} \log Z(x^n, w). \tag{3.29}$$

The derivative of $\log Z(x^n, w)$ is

$$\frac{\partial}{\partial w} \log Z(x^n, w) = \frac{1}{Z(x^n, w)} \frac{\partial}{\partial w} \sum_{y \in \mathcal{Y}} \exp(\langle w, \varphi(x^n, y)\rangle) \tag{3.30}$$

$$= \frac{1}{Z(x^n, w)} \sum_{y \in \mathcal{Y}} \exp(\langle w, \varphi(x^n, y)\rangle) \varphi(x^n, y) \tag{3.31}$$

$$= \sum_{y \in \mathcal{Y}} p(y|x^n, w) \varphi(x^n, y) \tag{3.32}$$

$$= \mathbb{E}_{y \sim p(y|x^n, w)} \left[\varphi(x^n, y) \right] \tag{3.33}$$

Substituting this into (3.29) we get

$$\nabla_w \mathcal{L}(w) = \sum_{n=1}^{N} \left[-\varphi(x^n, y^n) + \mathbb{E}_{y \sim p(y|x^n, w)} \left[\varphi(x^n, y) \right] \right] \tag{3.34}$$

It has been observed that maximizing the log-likelihood directly can lead to overfitting to the training set [25]. Therefore, we will impose some regularization on the weights and specifically we will use L_2

regularization which changes the negative log-likelihood to

$$\mathcal{L}^{\lambda}(w) = \lambda||w||^2 - \sum_{n=1}^{N}\langle w, \varphi(x^n, y^n)\rangle + \sum_{n=1}^{N}\log Z(x^n, w) \tag{3.35}$$

where λ is a non-negative regularization parameter. The gradient becomes

$$\nabla_w \mathcal{L}^{\lambda}(w) = 2\lambda w - \sum_{n=1}^{N}\left[\varphi(x^n, y^n) - \mathbb{E}_{y\sim p(y|x^n,w)}\left[\varphi(x^n, y)\right]\right]. \tag{3.36}$$

The use of regularization can also be motivated from a Bayesian perspective by imposing a prior on w and performing a MAP estimation of the weights. The posterior over the weights given the dataset $\mathcal{D}$ becomes

$$p(w|\mathcal{D}) = \frac{p(\mathcal{D}|w)p(w)}{p(\mathcal{D})} \tag{3.37}$$

$$= p(w)\frac{p(y^1, \ldots, y^N|x^1, \ldots, x^N, w)}{p(y^1, \ldots, y^N|x^1, \ldots, x^N)} \tag{3.38}$$

$$= p(w)\prod_{n=1}^{N}\frac{p(y^n|x^n, w)}{p(y^n|x^n)} \tag{3.39}$$

The MAP estimate of w is

$$w^* = \arg\max_{w\in\mathbb{R}^D} p(w|\mathcal{D}) \tag{3.40}$$

$$= \arg\max_{w\in\mathbb{R}^D} p(w)\prod_{n=1}^{N} p(y^n|x^n, w) \tag{3.41}$$

$$= \arg\max_{w\in\mathbb{R}^D}\left(\log p(w) + \sum_{n=1}^{N}\log p(y^n|x^n, w)\right) \tag{3.42}$$

$$= \arg\min_{w\in\mathbb{R}^D}\left(-\log p(w) - \sum_{n=1}^{N}\langle w, \varphi(x^n, y^n)\rangle + \sum_{n=1}^{N}\log Z(x^n, w)\right) \tag{3.43}$$

Imposing a Gaussian prior over w, $p(w) = \exp(-\frac{||w||^2}{2\sigma^2})$, we have

$$-\log p(w) = \frac{||w||^2}{2\sigma^2} \tag{3.44}$$

$$= \lambda ||w||^2 \tag{3.45}$$

which is exactly the regularizer we added in equation (3.35) where $\lambda = \frac{1}{2\sigma^2}$.

3.5 Numerical optimization

Since there does not exist a closed form solution to the optimization problem (3.24) we must resort to numerical optimization methods. Popular methods for numerical optimization are *gradient descent* methods [42] where the parameter space is searched iteratively in a certain descent direction δ_t in each iteration as follows:

$$w_{t+1} = w_t + \eta_t \delta_t \tag{3.46}$$

Many variants of the gradient descent algorithm exist and the most straight forward gradient descent method is called *steepest descent* [42], which chooses the descent direction to be the negative gradient. So let $g_t = \nabla_{w_t} \mathcal{L}(w_t)$ be the gradient of the negative log-likelihood, then

$$w_{t+1} = w_t - \eta_t g_t \tag{3.47}$$

Naïve steepest descent is known to have slow convergence rate and therefore second-order methods are often used instead [42]. These methods take curvature into account when computing the descent direction using the inverse Hessian H_t^{-1}. This gives us the following update step

$$w_{t+1} = w_t - \eta_t H_t^{-1} g_t \tag{3.48}$$

which is called the *Newton method*. The Newton method is shown to have a faster convergence rate compared to the steepest descent. However, it comes at the cost of increased computational complexity as the Hessian needs to be computed in each iteration. For large-scale problems with thousands or millions of variables, even storing the full Hessian may not be possible.

To amend both of these problems the *limited-memory Broyden-Fletcher-Goldfarb-Shanno* (LBFGS) method [40, 41] has been developed. In the original BFGS algorithm, the inverse Hessian is approximated by a matrix B_t which is computed from gradient information only. Therefore, we can avoid computing the actual Hessian. Let $s_t = w_{t+1} - w_t$ and $y_t = g_{t+1} - g_t$, then the BFGS method updates each iterate as follows:

$$\delta_t = -B_t g_t \tag{3.49}$$

$$w_{t+1} = w_t + \eta_t \delta_t \tag{3.50}$$

where

$$B_{t+1} = (I - \rho_t s_t y_t^T) B_t (I - \rho_t y_t s_t^T) + \rho_t s_t s_t^T \tag{3.51}$$

$$\rho_t = \frac{1}{y_t^T s_t} \tag{3.52}$$

The limited-memory version of BFGS stores a low-rank representation of the matrix in memory by computing it only from the past m gradients. Specifically, let $V_t = I - \rho_t y_t s_t^T$, then each iterate is computed as follows

$$\begin{aligned} B_{t+1} = {} & (V_t^T \cdots V_{t-m}^T) B_0 (V_{t-m} \cdots V_t) \\ & + \rho_{t-m} (V_t^T \cdots V_{t-m+1}^T) s_{t-m} s_{t-m}^T (V_{t-m+1} \cdots V_t) \\ & + \rho_{t-m+1} (V_t^T \cdots V_{t-m+2}^T) s_{t-m+1} s_{t-m+1}^T (V_{t-m+2} \cdots V_t) \\ & \vdots \\ & + \rho_t s_t s_t^T \end{aligned} \tag{3.53}$$

Having computed a descent direction we need to specify the step length η_t. This is done in the line search procedure that finds the η_t such that it satisfies the following conditions:

$$\mathcal{L}(w_t + \eta_t \delta_t) \leq \mathcal{L}(w_t) + \alpha \eta_t g_t^T \delta_t \tag{3.54}$$

$$g(w_t + \eta_t \delta_t)^T \delta_t \geq \beta g_t^T \delta_t \tag{3.55}$$

called the *Wolfe conditions*, where typically $\alpha = 10^{-4}$ and $\beta = 0.9$ [42]. The conditions ensure that the selected step length brings sufficient decrease to the objective function (3.54) and that the step taken is

not too small (3.55). Note than when performing line search with the Wolfe conditions both the objective function and the gradient must be evaluated for every possible η_t. As we will see in the next section evaluating the gradient is very time consuming and therefore evaluating it for every step length during the line search procedure is not tractable. One way to alleviate this problem is to run the line search only based on the first condition (often called the *Armijo condition*), however, then we no longer have any convergence guarantees and we may end up taking far more iterations to converge and lose performance gained in the line search. It is beyond the scope of this thesis to discuss convergence guarantees of the LBFGS method with different line search methods and we choose to use the Wolfe conditions in order to ensure good convergence rates.

3.6 Computational complexity

Both the negative log-likelihood and its gradient is going to be evaluated many times during the optimization procedure. Exactly how many iterations are needed for convergence is hard to tell and depends heavily on the starting point w_0 and the regularizer λ. Having a larger value of λ means that the objective function will look more like a quadratic function and since LBFGS approximates the function at each iteration by a quadratic function it will converge faster. However, just one single iteration of the LBFGS requires computing both the log-likelihood and the gradient at least once which can be intractable in itself.

To see that learning the parameters for the CRF model can be intractable, let us take a look at the computational complexity of evaluating the log-likelihood and the gradient. Let N be the number of training examples, P be the number of feature points in each image, $|\mathcal{Y}|$ be the number of possible bounding boxes for a given image, D the dimensionality of w and K the number iterations of LBFGS to reach optimum. Computing the regularized negative log-likelihood (3.35) requires three steps: 1) computing the regularizer, 2) computing the dot product of w and the feature map for all images and 3) computing the log partition function for all images. The regularizer takes $O(D)$ operations to compute and the dot product of w with $\varphi(x, y)$ corresponds

to four lookups in the integral image as explained in Chapter 2 and since this is done for each image we have $O(N)$ operations for the dot products in addition to the $O(NP + NH^2)$ operations to initialize the integral image. Finally, computing the log partition function requires summing over all bounding boxes for each image. Assume we have an image of dimension $H \times W$. Since we have $y^n = (l, t, r, b)$ where $t, b < H$ and $l, r < W$ and $l < r$ and $t < b$ we can rewrite the computation of $Z(x^n, w)$ as follows.

$$Z(x^n, w) = \sum_{y \in \mathcal{Y}} \exp \langle w, \varphi(x^n, y^n) \rangle \tag{3.56}$$

$$= \sum_{l=0}^{W} \sum_{t=0}^{H} \sum_{r=l+1}^{W} \sum_{b=t+1}^{H} \exp \left(\langle w, \varphi(x^n, l, t, r, b) \rangle \right) \tag{3.57}$$

The number of possible bounding boxes is given by the formula

$$|\mathcal{Y}| = \left(\frac{H(H+1)}{2} \right) \left(\frac{W(W+1)}{2} \right) \tag{3.58}$$

where the first term corresponds to the number of possible (t, b) pairs and the second term corresponds to the number of (l, r) pairs. For an image of size 640 $\times$ 480 which is not uncommon in the PASCAL dataset, the number of possible bounding boxes is more than 23 billion! To simplify the analysis, we will assume images of dimension $H \times H$ in which case the number of bounding boxes is $O(H^4)$. Thus, the total computation time is $O(D + N + NH^4)$:

$$\mathcal{L}^\lambda(w) = \underbrace{\lambda ||w||^2}_{O(D)} + \underbrace{\sum_{n=1}^{N} \langle w, \varphi(x^n, y^n) \rangle}_{O(N)} - \underbrace{\sum_{n=1}^{N} \log Z(x^n, w)}_{O(NH^4)} \tag{3.59}$$

plus an additional $O(NP + NH^2)$ for computing the integral image.

Computing the gradient similarly consists of three steps: First, computing the derivative of the regularizer takes $O(D)$ operations. Using the integral histogram technique described above, computing the feature map takes four histogram lookups per image, that is $O(ND)$. Finally, the expectation is the most time critical part of the gradient evaluation. We saw in equation (3.33) that the expectation can be

written as

$$\mathbb{E}_{y\sim p(y|x^n,w)}\left[\varphi(x^n,y)\right] = \frac{1}{Z(x^n,w)}\sum_{y\in\mathcal{Y}}\exp(\langle w,\varphi(x^n,y^n)\rangle)\varphi(x^n,y). \tag{3.60}$$

Thus, we see that the partition function can be computed outside the expectation which we saw requires a constant operation over $O(H^4)$ boxes. Additionally, computing the expectation we need another sum over the total number of bounding boxes. The terms within the sum have a computational cost of $O(D)$ because computing $\langle w,\varphi(x^n,y^n)\rangle$ is four lookups and $\varphi(x^n,y^n)$ is $O(D)$. This is done for every image which gives us the total computational cost of $O(D+ND+NH^4+NDH^4)$:

$$\nabla_w\mathcal{L}^\lambda(w) = \underbrace{2\lambda w}_{O(D)} + \underbrace{\sum_{n=1}^{N}\varphi(x^n,y^n)}_{O(ND)} - \underbrace{\sum_{n=1}^{N}\mathbb{E}_{y\sim p(y|x^n,w)}\left[\varphi(x^n,y)\right]}_{O(NH^4+NDH^4)} \tag{3.61}$$

In addition we need to compute the integral images. Furthermore, we use a method similar to the integral image to look up a feature map in $O(D)$ operations. This is called an integral histogram and the computation of this for all images requires $O(NP+NDH^2)$ operations.

Thus, the computational complexity for running LBFGS for K iterations is $O(K(NP+NDH^2))$ for computing the integral images and integral histograms, $O(K(D+N+NH^4))$ for computing the log-likelihoods and $O(K(D+ND+NH^4+NDH^4)))$ for computing the gradients. Overall this reduces to $O(KNDH^4)$ by ignoring all the lower order terms. To get a rough idea of the number of operations required here is the approximate magnitudes of the four variables:

$$K \sim [20,100] \tag{3.62}$$

$$N \sim [90,130] \tag{3.63}$$

$$D = 3000 \tag{3.64}$$

$$H \sim [300,600] \tag{3.65}$$

So clearly, the dominant term in the time complexity is $O(H^4)$ and since computing the feature map for one image is $O(D)$ operations using

the integral histogram, the $O(NDH^4)$ term, although still polynomial, makes computing the gradient infeasible in practice for even medium-sized images and medium-sized datasets.

3.7 Implementation details

When implementing the optimization algorithms described in this and the next two chapters, we obviously need to compute both the negative log-likelihood as well as the gradient several times and as shown above it is crucial to make these calculations efficient and numerically stable.

The first trick we use is for numerical stability purposes. There are two issues in computing (3.15) directly. First of all, when performing line search in the LBFGS algorithm the weight vector w may jump to large values causing numerical overflow when computing $\exp(\langle w, \varphi(x, y)\rangle)$. Secondly, since the partition function $Z(x, w)$ is a sum over all possible bounding boxes with these weights it, too, will cause a numerical overflow. To amend this problem we note that we can rewrite the distribution as follows:

$$p(y|x, w) = \frac{1}{Z(x, w)} \exp\left(\langle w, \varphi(x, y)\rangle\right) \tag{3.66}$$

$$= \exp\left(\log\left(\frac{1}{Z(x, w)} \exp\left(\langle w, \varphi(x, y)\rangle\right)\right)\right) \tag{3.67}$$

$$= \exp\left(\langle w, \varphi(x, y)\rangle - \log Z(x, w)\right) \tag{3.68}$$

Because we take the logarithm of the partition function we can compute it for much larger values of w and subtracting this from the dot product in the exponent ensures that the exponential evaluates to something between 0 and 1. This is a classical trick and the CRF model is often written directly in this form [61].

Now we need to evaluate the log partition function:

$$\log Z(x, w) = \log \sum_{y \in \mathcal{Y}} \exp\left(\langle w, \varphi(x, y)\rangle\right) \tag{3.69}$$

This has the form of the log of a sum of exponentials. This particular

form can be rewritten using the log-sum-exp trick as follows [44]:

$$\log Z(x, w) = \log \sum_{y \in \mathcal{Y}} \exp\left(\langle w, \varphi(x, y)\rangle\right) \tag{3.70}$$

$$= \log \sum_{y \in \mathcal{Y}} \exp\left(\langle w, \varphi(x, y)\rangle\right) \frac{\exp(\alpha)}{\exp(\alpha)} \tag{3.71}$$

$$= \log \left(\exp(\alpha) \sum_{y \in \mathcal{Y}} \exp\left(\langle w, \varphi(x, y)\rangle - \alpha\right) \right) \tag{3.72}$$

$$= \alpha + \log \sum_{y \in \mathcal{Y}} \exp\left(\langle w, \varphi(x, y)\rangle - \alpha\right) \tag{3.73}$$

Choosing $\alpha = \max_{y \in \mathcal{Y}} \langle w, \varphi(x, y)\rangle$, we see that the exponent will always be non-positive and computing $\log Z(x, w)$ becomes numerically stable. This comes at the added computational cost of going through all y twice, first to determine α and then to compute the sum. To speed this up one could use the ESS algorithm [39] to efficiently compute α.

Besides numerical stability we also need to address computational efficiency. In Chapter 2 we already saw how computing the dot product can be performed by four lookups in the integral image. This gives significant speedup compared to a naïve computation of the dot product but comes at the expense of having to compute the integral image for each x. However, the number of times we need to evaluate the probability of a bounding box (for example when computing the expectation in the gradient) far outweighs the number of times we need to recompute the integral image.

In the previous section on computational complexity we saw that the number of bounding boxes makes computation of the gradient especially time consuming. To cut down the number of possible bounding boxes we quantize the coordinates of bounding boxes with a predefined step size.

Finally, we note that the computation of the log-likelihood and the gradient is embarrassingly parallel, since we can compute the contribution of each image in the dataset separately and then sum them up afterwards. For a dataset of size N, this means that we can get almost N times speedup by computing the terms within the sum on N different processors. We have implemented a parallel version of MCL that

does this.

3.8 Summary

We introduced conditional random fields as a way of modeling the object localization problem. We showed how to perform sampling in these models with a Gibbs sampler and we showed how to learn the parameters of the model by maximum conditional likelihood (MCL) estimation. Furthermore, we showed how to deal with numerical stability and efficient implementation of the formulas. Unfortunately, even with careful code optimization, direct optimization of the likelihood is intractable for medium-sized images and medium-sized datasets and in the next two chapters we will discuss two types of approximations.

4

Approximating the gradient

In this chapter we look at the type of approximations that optimizes the real objective function but uses approximate gradients during the gradient descent optimization procedure. We discuss two stochastic gradient methods, stochastic gradient descent and contrastive divergence, and we show the time complexity for both methods.

4.1 Introduction

As described in Chapter 3 computing the gradient of the objective function is often most time consuming if not intractable. Therefore, it is worthwhile to avoid as many gradient evaluations as possible and, if possible, speed up the gradient evaluations. We saw that the two major problems with computing the gradient is that it requires summing over all input images as well summing over all outputs for each image.

For large datasets, summing over all data points can make the optimization procedure intractable even if the computation per data point is tractable. One solution to this problem would be to simply perform the parameter learning on a subset of the dataset. However, doing so would lose valuable information present in the data points not seen

during training. Stochastic gradient descent has proven successful in learning algorithms since it traverses the entire dataset while updating the weights after evaluating only a single data point.

Computing the gradient also involves computing the expectation over the model distribution for each input image. For large images this easily becomes intractable in itself. Besides downsampling the images so as to cut down the number of possible bounding boxes we can use sampling techniques to approximate the expectation by computing the mean over a number of sampled bounding boxes. However, sampling from the model distribution $p(y|x, w)$ using Gibbs sampling is also computationally expensive since we need to take many steps of the Markov chain to obtain a single sample. Luckily, it turns out that using a single sample obtained from running one step of the Markov chain gives a close enough approximation of the expectation to be used in a stochastic gradient descent optimization. This is called contrastive divergence learning, which we will describe in Section 4.3.

4.2 Stochastic gradient descent

We saw in the previous chapter that running a second order gradient descent algorithm with a full line search requires many log-likelihood and gradient evaluations, both of which sum over all training examples. The gradient is especially time consuming because of the expectation that needs to be computed for each image. One way to deal with this is to use the crucial insight by Bottou and Bousquet [9] and note that since we are already approximating the data distribution with some model distribution we do not necessarily have to find the exact optimum of the log-likelihood and we can stop the optimization procedure before convergence.

Instead of using the expensive LBFGS procedure to optimize the log-likelihood one can resort to stochastic approximation of the gradient. This leads to the *stochastic gradient descent* (SGD) optimization procedure. The approximation in SGD consists of computing the gradient for only one single image picked randomly from the dataset. Let

(x^n, y^n) be a randomly chosen data point, then the gradient becomes

$$\tilde{\nabla}_w^{(x^n,y^n)}\mathcal{L}^\lambda(w) = 2\lambda w - \varphi(x^n, y^n) + \mathbb{E}_{y\sim p(y|x^n,w)}\left[\varphi(x^n, y)\right] \tag{4.1}$$

Furthermore, by not using a line search procedure for finding the correct step size we can avoid evaluating the log-likelihood as well. Instead we choose a fixed *learning rate* η_t for each iteration t. This gives us the following stochastic gradient descent update rule:

$$w_{t+1} = w_t - \eta_t \tilde{\nabla}_{w_t}^{(x^n,y^n)}\mathcal{L}^\lambda(w_t) \tag{4.2}$$

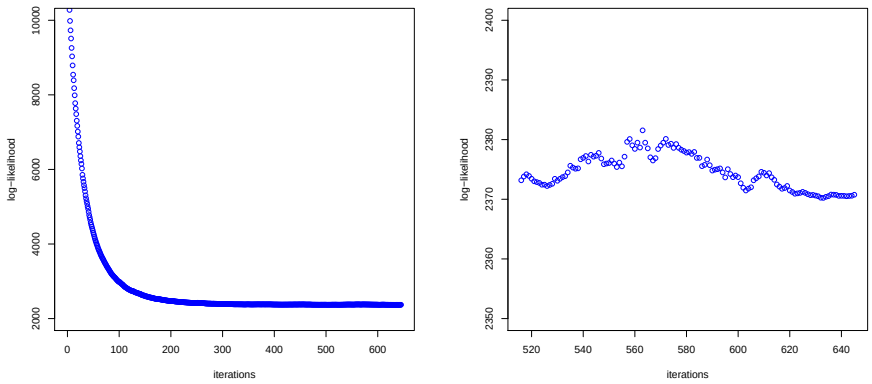
We see that each iteration of the SGD procedure requires only three simple steps:

(1) Pick a random data point (x^n, y^n)
(2) Compute $\tilde{\nabla}_{w_t}^{(x^n,y^n)}\mathcal{L}^\lambda(w_t)$ by (4.1)
(3) Compute new weights w_{t+1} by (4.2)

Comparing with the real gradient (3.34) we see that the only thing that has changed is that we no longer sum over all images. Therefore, the computational complexity of the gradient is $O(DH^4)$ and updating the weights requires an additional $O(D)$ operations in addition to the $O(P+DH^2)$ operations required for computing the integral histogram. This is an improvement over the LBFGS, especially if the dataset is very large. But because the gradient is only an approximation of the real gradient, it will require more iterations before convergence. Indeed, the stochastic gradient does not necessarily point in the direction of the real gradient and may in fact point in the opposite direction. However, with a properly chosen learning rate convergence is guaranteed [7, 9]. In Figure 4.1 we see the objective value for five stochastic gradient descent iterations. We see how the function value oscillates but decreases nonetheless.

In order to guarantee convergence we have to decide on a reasonable learning rate η_t. We choose the learning rate recommended by Bordes et al. [7] for first order SGD, namely

$$\eta_t = \frac{1}{\alpha(t + t_0)} \tag{4.3}$$



(a) (b)

Figure 4.1 Oscillations of the stochastic gradient descent. (a) shows five epochs (passes through the dataset) of SGD on the horse images from the PASCAL dataset. Clearly, the log-likelihood decreases during training especially in the first couple of epochs. In (b) we see the iterations of the fifth epoch and we see that the approximate gradient of SGD makes the log-likelihood oscillate between iterations.

This gives us two parameters to choose: α which determines the overall magnitude of the learning rate and t_0 which determines how fast the learning rate decreases with each iteration. By experimentation we choose α to be equal to the regularization constant λ and choose t_0 by running one pass[1] over the training set with different values of t_0 and choose the value that decreases the negative log-likelihood the most. This trick is also used by Bottou's CRF SGD code[2] as well as by Bordes et al. for SGD-QN [7] where they use only a small subset of the dataset. Since we only look at images in the training set that include the desired object, we already train on only a subset of the entire training set and therefore we choose to run one full pass for each t_0 we try.

There exists two variants of SGD, second-order SGD [61] and averaged SGD [65], that can be shown to have faster convergence for a single pass through the dataset. Since the limitation of our optimiza-

[1] a pass through the dataset corresponds to N iterations of stochastic gradient descent where each image in the dataset is used exactly once.

[2] the `crfsgd` source code can be obtained here: `http://leon.bottou.org/projects/sgd`

tion is not the number of images but the output space we implement the simple standard SGD. However, we do perform averaging of the final pass through the dataset. This and further implementation details will be discussed in Section 4.4.

4.3 Contrastive divergence

SGD still needs to compute the expectation over the model distribution in (4.1) which, in itself, can be intractable. In addition to the stochastic gradient approximation we can approximate this expectation by the sample mean of S samples obtained from the model distribution. As we saw in Section 3.3 we can easily obtain samples from the model distribution using the Gibbs sampler and thus we can approximate the log-likelihood gradient from a set of samples $\{\tilde{y}^{(1,k)}, \tilde{y}^{(2,k)}, \ldots, \tilde{y}^{(S,k)}\}$ where each sample is computed by taking k steps in the Gibbs chain:

$$\hat{\nabla}_w^{(x^n,y^n)} \mathcal{L}^\lambda(w) = 2\lambda w - \varphi(x^n, y^n) + \frac{1}{S} \sum_{s=1}^{S} \varphi(x^n, \tilde{y}^{(s,k)}) \tag{4.4}$$

Going to the extreme we can use just a single sample $\tilde{y}^{(k)}$ which is initialized at the ground truth for approximating the expectation. This gives us the *contrastive divergence* (CD) gradient initially proposed by Hinton [28]:

$$\hat{\nabla}_w^{(x^n,y^n)} \mathcal{L}^\lambda(w) = 2\lambda w - \varphi(x^n, y^n) + \varphi(x^n, \tilde{y}^{(k)}) \tag{4.5}$$

Obtaining a sample from the model distribution $p(y|x, w)$ requires a long run of the Gibbs chain in order for the sample to be independent of the ground truth. Luckily, it has been shown [2, 10, 28] that the bias that occurs when not running the Gibbs chain long enough is small in practice. Therefore, it makes sense to only run the Gibbs chain for a small number of steps. Even the extreme case where $k = 1$, CD is shown to give reasonable results in practice [10].

The contrastive divergence was originally motivated by minimizing the difference between two Kullback-Leibler divergences [28]. Given a Gibbs sampler that is initialized at the data distribution $d(y|x)$ and has $p(y|x, w)$ as its stationary distribution, let $p_k(y|x, w)$ be a distribution from which the k'th sample in the Gibbs chain is obtained. Then we

have $p_0(y|x) = d(y|x)$ and $p_\infty(y|x,w) = p(y|x,w)$ and the KL divergence between p_∞ and p_0 is (as we saw in Section 3.4)

$$\begin{aligned}\mathrm{KL}(p_\infty||p_0) &= \sum_{y\in\mathcal{Y}} p_0(y|x)\log p_0(y|x) - \sum_{y\in\mathcal{Y}} p_0(y|x)\log p_\infty(y|x,w) \\ &= -H(p_0) - \sum_{y\in\mathcal{Y}} p_0(y|x)\log p_\infty(y|x,w) \end{aligned} \tag{4.6}$$

where $H(p_0)$ is the Shannon entropy. Contrastive divergence learning then minimizes the following objective:

$$\text{CD-}k = \mathrm{KL}(p_\infty||p_0) - \mathrm{KL}(p_\infty||p_k) \tag{4.7}$$

The derivative of this with respect to w is

$$\begin{aligned}\frac{\partial}{\partial w}\text{CD-}k &= -\sum_{y\in\mathcal{Y}} p_0(y|x)\frac{\partial \log p_\infty(y|x,w)}{\partial w} \\ &\quad + \sum_{y\in\mathcal{Y}} \frac{\partial}{\partial w}\left(p_k(y|x,w)\log p_\infty(y|x,w)\right) \\ &= -\sum_{y\in\mathcal{Y}} p_0(y|x)\frac{\partial \log p_\infty(y|x,w)}{\partial w} \\ &\quad + \sum_{y\in\mathcal{Y}} p_k(y|x,w)\frac{\partial \log p_\infty(y|x,w)}{\partial w} \\ &\quad + \sum_{y\in\mathcal{Y}} \frac{\partial p_k(y|x,w)}{\partial w}\log p_\infty(y|x,w)\end{aligned} \tag{4.8}$$

The third term has been proven to be small in practice [2, 28] which justifies ignoring it. Furthermore, approximating the sums by a single sample from the data distribution y^n and a single sample from

$p_k(y|x,w)$ gives us

$$\begin{aligned}\frac{\partial}{\partial w}\text{CD-}k \approx &- \frac{\partial \log p_\infty(y^n|x^n,w)}{\partial w} \\ &+ \frac{\partial \log p_\infty(\tilde{y}^{(k)}|x^n,w)}{\partial w} \\ = &- \varphi(x^n,y^n) + \frac{\partial \log Z(x^n,w)}{\partial w} \\ &+ \varphi(x^n,\tilde{y}^{(k)}) - \frac{\partial \log Z(x^n,w)}{\partial w} \\ = &- \varphi(x^n,y^n) + \varphi(x^n,\tilde{y}^{(k)})\end{aligned} \tag{4.9}$$

which is exactly the CD gradient we obtained in Equation (4.5) without the regularization term. An intuitive explanation for the contrastive divergence objective function is that since we want the model distribution to be as close to the data distribution as possible, the distribution from which the k'th sample is obtained in the Gibbs chain should also be close to the data distribution.

Bengio and Delalleau [2] later showed that the CD gradient is actually also a biased estimator of the log-likelihood gradient. CD has been thoroughly investigated for training a special kind of bipartite MRF called restricted Boltzmann machines (RBMs) [2, 10, 20, 21, 22, 29] but has also been used for training CRFs [27, 34, 45]. Since CD is a stochastic gradient approximation of the log-likelihood gradient, we can perform stochastic gradient descent as above:

$$w_{t+1} = w_t - \eta_t \hat{\nabla}^{(x^n,y^n)}_{w_t} \mathcal{L}^\lambda(w_t) \tag{4.10}$$

This gives us the following outline of the contrastive divergence learning method:

(1) Pick a random data point (x^n, y^n)
(2) Sample $\tilde{y}^{(k)}$ from $p(y|x,w)$ by running k steps of the Gibbs chain starting at y^n
(3) Compute $\hat{\nabla}^{(x^n,y^n)}_{w_t} \mathcal{L}^\lambda(w_t)$ by (4.5)
(4) Compute new weights w_{t+1} by (4.10)

The computational complexity of the CD learning algorithm now depends on the Gibbs sampler. One Gibbs step requires computing the

cumulative histogram for each of the four variables, which takes $O(H)$ operations and we do this for k steps. Therefore, the computational complexity of the CD gradient becomes $O(DH)$ plus the computation of the integral histogram which is $O(P + DH^2)$.

We can generalize the sampled gradient (4.4) by summing over a subset of the data points instead of just one. Let this subset have size $N' \leq N$. This gives us a whole family of gradient approximations parameterized by N', S and k, the batch size, the number of samples and the number of Gibbs chain steps, respectively:

$$\hat{\nabla}_w^{N',S,k} \mathcal{L}^\lambda(w) = 2\lambda w - \sum_{n=1}^{N'} \left[\varphi(x^n, y^n) - \frac{1}{S} \sum_{s=1}^{S} \varphi(x^n, \tilde{y}^{(s,k)}) \right] \quad (4.11)$$

We see that this form of the gradient generalizes both the stochastic gradient and contrastive divergence gradient as well as the real log-likelihood gradient. Setting $N' = N$ and letting $S \to \infty$ and $k \to \infty$ we recover the real log-likelihood gradient (3.36). Setting $N' = 1$ we obtain the stochastic gradient (4.1). The CD-k gradient (4.5) is obtained by setting $N' = 1$ and $S = 1$. We investigate only the simplest CD method with $N' = 1$, $S = 1$ and $k = 1$ but the effects of adjusting the three parameters would be an interesting study.

4.4 Implementation details

Since both the stochastic gradient and the contrastive divergence gradient can be formulated as a stochastic approximation to the log-likelihood gradient the implementation of the descent algorithm is identical. In fact, we view CD as simply a stochastic gradient descent where the gradient is computed as (4.5) instead of (4.1). We will now discuss implementation details in terms of SGD but everything applies to CD learning as well. For more details on practical CD implementation concerns, see Hintons guide for training RBMs [29].

When running SGD we want to use the whole training set and therefore, instead of picking training examples randomly, we go through the training examples until all images have been used for computing the gradient. We call such a pass through the training set an *epoch*. To avoid artifacts due to the ordering of the training examples, we

shuffle the training set after each epoch. Note that one epoch of the SGD takes approximately as long as one single iteration of LBFGS but in this same time SGD will have run N iterations and will hopefully have reached a better weight vector than the single LBFGS step. This depends on the learning rate though.

Although the SGD is simple in theory, making it run and converge in due time requires some engineering of the parameters. Most importantly is how to choose the learning rate η_t. We apply the following strategy inspired by Bottou: We choose an initially low value of the learning rate and run an entire epoch of SGD. We repeat this for increasing values of η_0 until the objective begins to increase instead of decrease. We then pick the learning rate that decreased the objective the most and compute t_0 as

$$t_0 = \frac{1}{\lambda \eta_0} \tag{4.12}$$

and from this we can start the real SGD procedure. This is a heuristic for setting t_0 which has proven to work well in practice [7, 9].

Because of the oscillations in objective between each iteration in the optimization algorithm, the last obtained weight vector may not be the best solution. A popular and simple to implement variation of the standard SGD method is the averaged SGD in which the result of the method is the average of all weight vectors obtained during the SGD procedure [65]. Instead of computing the running average of all obtained weight vectors we average only the iterates of the last epoch. This is similar to what is proposed by Rahklin et al. [49] who show that this leads to faster convergence rates for strongly convex objective functions. In this paper they only run a single epoch and average the last qN iterations for some $q \in [0, 1]$. Averaging the weights of the last epoch makes sense because the SGD is closer to convergence in the final epoch and therefore these iterates are closer to the optimum than in the previous epochs.

4.5 Summary

We described two methods for optimizing the log-likelihood objective function based on stochastic gradient approximations. The first

method, stochastic gradient descent, updates the weights after having computed the gradient from just a single training set example. This makes learning tractable for large training sets where computing the expectation for one image is tractable. The second approach, contrastive divergence learning, in addition approximates the expectation in the gradient from a single bounding box sample obtained from running few steps of a Gibbs sampler. This makes computing the gradient tractable for large output spaces.

5

Approximating the log-likelihood

In this chapter we explore how we can accelerate the finding of good weights for the conditional random field model by approximating the log-likelihood by alternative objectives during training. Specifically, we present parameter learning by maximum pseudolikelihood and derive formulas that allow piecewise training.

5.1 Introduction

Instead of solving the real maximum conditional likelihood problem, an alternative is to optimize an approximation of the likelihood that is actually computationally tractable. We will explore this approach in this chapter by considering two approximations of the log-likelihood with respect to the localization problem: the pseudolikelihood in Section 5.2 and piecewise training in Section 5.3. It is important to note that the approximation of the log-likelihood only takes place during training; at test time the resulting weights are applied in the true model as if they had been trained using maximum conditional likelihood.

5.2 Pseudolikelihood

Pseudolikelihood (PL) was invented by Besag [3, 4] and it approximates the true likelihood $p(y|x, w)$ with respect to the weights by assuming that every output node $y_s, s = 1, \ldots, M$ depends only on observed quantities and therefore the likelihood can be computed as a product of per-node conditional probabilities. Under this assumption the optimal value of each y_s can be found by independent maximization. During training both x and y are observed, so we can define the pseudolikelihood of a data pair (x^n, y^n) by

$$p_{PL}(y^n|x^n, w) = \prod_{s=1}^{M} p_{PL}(y_s^n|y_{\neg s}^n, x^n, w), \tag{5.1}$$

where

$$y_{\neg s}^n = (y_1^n, \ldots, y_{s-1}^n, y_{s+1}^n, \ldots, y_M^n) \tag{5.2}$$

and the per-node conditional probabilities $p_{PL}(y_s^n|y_{\neg s}^n, x^n, w)$ are given by

$$p_{PL}(y_s^n|y_{\neg s}^n, x^n, w) = \frac{1}{Z_s(x^n, y_{\neg s}^n, w)} \exp(-E(x^n, y^n, w)), \tag{5.3}$$

with the partition function

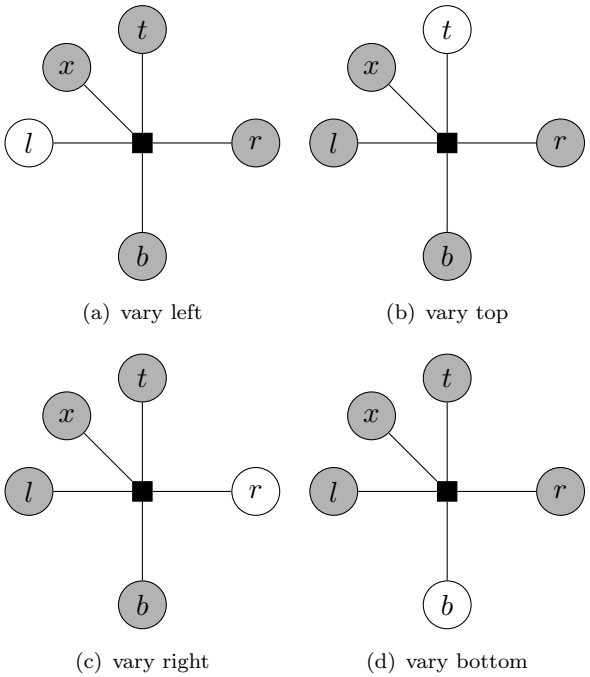
$$Z_s(x^n, y_{\neg s}^n, w) = \sum_{y_s \in \mathcal{Y}_s} \exp(-E(x^n, y_{\neg s}^n, y_s, w)) \tag{5.4}$$

and the energy function

$$E(x^n, y^n, w) = -\langle w, \varphi(x^n, y^n)\rangle \tag{5.5}$$

In the problem of object localization we have the four output nodes *left*, *top*, *right*, and *bottom* that correspond to the sides of the bounding box, so here we understand the assumption behind pseudolikelihood to mean that we in turn keep the position of three of the four sides constant while we vary the position of the remaining side. See in Figure 5.1 the different graph structures reflecting these cases.

As for maximum conditional likelihood learning in Section 3.4, we assume that we are given a dataset $\mathcal{D} = \{(x^n, y^n)|n = 1, 2, \ldots, N\}$ of



(a) vary left (b) vary top

(c) vary right (d) vary bottom

Figure 5.1 Probabilistic graphical model of the bounding box for each of the four cases of pseudolikelihood.

independent and identically distributed data pairs which then has the pseudolikelihood

$$p_{PL}(y^1, \ldots, y^N | x^1, \ldots, x^N, w) = \prod_{n=1}^{N} p_{PL}(y^n | x^n, w) \tag{5.6}$$

and so we can find the approximate optimal weights by maximum pseudolikelihood estimation over the dataset

$$\begin{aligned} w_{PL}^* &= \arg\max_{w \in \mathbb{R}^D} p_{PL}(y^1, \ldots, y^N | x^1, \ldots, x^N, w) \\ &= \arg\max_{w \in \mathbb{R}^D} \prod_{n=1}^{N} p_{PL}(y^n | x^n, w) \\ &= \arg\min_{w \in \mathbb{R}^D} \mathcal{L}_{PL}(w) \end{aligned} \tag{5.7}$$

where the negative log-pseudolikelihood $\mathcal{L}_{PL}(w)$ can be expanded as

$$\begin{aligned}\mathcal{L}_{PL}(w) &= -\log\left(\prod_{n=1}^{N} p_{PL}(y^n|x^n,w)\right)\\ &= -\log\left(\prod_{n=1}^{N}\prod_{s=1}^{M} p_{PL}(y_s^n|y_{\neg s}^n,x^n,w)\right)\\ &= -\log\left(\prod_{n=1}^{N}\prod_{s=1}^{M} \frac{\exp(-E(x^n,y^n,w))}{Z_s(x^n,y_{\neg s}^n,w)}\right)\\ &= \sum_{n=1}^{N}\sum_{s=1}^{M}\left[E(x^n,y^n,w)+\log Z_s(x^n,y_{\neg s}^n,w)\right].\end{aligned} \tag{5.8}$$

By writing out the energy function, we get

$$\begin{aligned}\mathcal{L}_{PL}(w) &= \sum_{n=1}^{N}\sum_{s=1}^{M}\left[-\langle w,\varphi(x^n,y^n)\rangle+\log Z_s(x^n,y_{\neg s}^n,w)\right]\\ &= -M\sum_{n=1}^{N}\langle w,\varphi(x^n,y^n)\rangle+\sum_{n=1}^{N}\sum_{s=1}^{M}\log Z_s(x^n,y_{\neg s}^n,w).\end{aligned} \tag{5.9}$$

Minimizing $\mathcal{L}_{PL}(w)$, however, is prone to overfitting so we will add an L_2-regularizer and instead minimize the regularized negative log-pseudolikelihood $\mathcal{L}_{PL}^{\lambda}$

$$\mathcal{L}_{PL}^{\lambda}(w) = \lambda||w||^2 - M\sum_{n=1}^{N}\langle w,\varphi(x^n,y^n)\rangle+\sum_{n=1}^{N}\sum_{s=1}^{M}\log Z_s(x^n,y_{\neg s}^n,w). \tag{5.10}$$

The gradient of $\mathcal{L}_{PL}(w)$ with respect to the weights can be derived as follows

$$\begin{aligned}\nabla_w\mathcal{L}_{PL}(w) &= \frac{\partial}{\partial w}\left(-M\sum_{n=1}^{N}\langle w,\varphi(x^n,y^n)\rangle+\sum_{n=1}^{N}\sum_{s=1}^{M}\log Z_s(x^n,y_{\neg s}^n,w)\right)\\ &= -M\sum_{n=1}^{N}\varphi(x^n,y^n)+\sum_{n=1}^{N}\sum_{s=1}^{M}\frac{\partial}{\partial w}\log Z_s(x^n,y_{\neg s}^n,w),\end{aligned} \tag{5.11}$$

where the gradient of the log-partition function with respect to the weights is

$$\begin{aligned}\frac{\partial}{\partial w}\log Z_s(x^n, y^n_{\neg s}, w) &= \frac{1}{Z_s(x^n, y^n_{\neg s}, w)}\frac{\partial}{\partial w}Z_s(x^n, y^n_{\neg s}, w)\\ &= \sum_{y_s\in\mathcal{Y}_s}\frac{\exp(\langle w, \varphi(x^n, y^n_{\neg s}, y_s)\rangle)}{Z_s(x^n, y^n_{\neg s}, w)}\varphi(x^n, y^n_{\neg s}, y_s)\\ &= \sum_{y_s\in\mathcal{Y}_s} p_{PL}(y_s|y^n_{\neg s}, x^n, w)\varphi(x^n, y^n_{\neg s}, y_s)\\ &= \mathbb{E}_{y_s\sim p_{PL}(y_s|y^n_{\neg s}, x^n, w)}\left[\varphi(x^n, y^n_{\neg s}, y_s)\right].\end{aligned} \tag{5.12}$$

We end up with

$$\begin{aligned}\nabla_w\mathcal{L}_{PL}(w) = &-M\sum_{n=1}^{N}\varphi(x^n, y^n)\\ &+\sum_{n=1}^{N}\sum_{s=1}^{M}\mathbb{E}_{y_s\sim p_{PL}(y_s|y^n_{\neg s}, x^n, w)}\left[\varphi(x^n, y^n_{\neg s}, y_s)\right].\end{aligned} \tag{5.13}$$

The gradient of the regularized negative log-pseudolikelihood $\mathcal{L}^{\lambda}_{PL}$ is likewise

$$\begin{aligned}\nabla_w\mathcal{L}^{\lambda}_{PL}(w) = 2\lambda w &- M\sum_{n=1}^{N}\varphi(x^n, y^n)\\ &+\sum_{n=1}^{N}\sum_{s=1}^{M}\mathbb{E}_{y_s\sim p_{PL}(y_s|y^n_{\neg s}, x^n, w)}\left[\varphi(x^n, y^n_{\neg s}, y_s)\right].\end{aligned} \tag{5.14}$$

It can be shown that maximum pseudolikelihood is consistent, meaning that it will obtain the same solution as real maximum likelihood [24, 31, 64]. An intuition given by Sutton and McCallum [54] is that pseudolikelihood attempts to match all of the model conditional distributions to those of the empirical distribution. If it succeeds in matching them all exactly and assuming that the model distribution includes the true distribution, then in the limit of infinite data, the empirical conditional distributions will equal the true conditional distributions and so the Gibbs sampler of the model will be the same as the Gibbs sampler of the true distribution.

5.2.1 Computational complexity

As in Section 3.6, we denote by N the number of training examples, P the number of feature points in an image, H the height of an image, and D the dimensionality of w. Computing the regularized negative log-pseudolikelihood (5.10) consists of three steps: 1) computing the regularizer, 2) computing the dot product of w with the bag of words for all images followed by a multiplication of the number of bounding box sides, and 3) computing the log-partition function for all sides of the bounding box and for all images. The regularizer takes $O(D)$ operations to compute. The integral image as described in Chapter 2 allows us to compute the dot product in constant time, and doing so for all images is $O(N)$ operations. The log-partition function is a sum of lookups in the integral image with the number of lookups corresponding to the number of positions that a single side of the bounding box can take. We have to do this for all images, so this is accomplished in $O(NH)$ operations. This gives us the complexity $O(D + N + NH)$:

$$\mathcal{L}_{PL}^{\lambda}(w) = \underbrace{\lambda||w||^2}_{O(D)} \underbrace{- M \sum_{n=1}^{N} \langle w, \varphi(x^n, y^n) \rangle}_{O(N)} + \underbrace{\sum_{n=1}^{N} \sum_{s=1}^{M} \log Z_s(x^n, y^n_{\neg s}, w)}_{O(NH)} \tag{5.15}$$

plus the additional term of $O(NP + NH^2)$ for setting up the integral images.

The complexity of the gradient of the regularized negative log-pseudolikelihood (5.14) can be shown in a similar fashion. Computing the gradient consists of the three steps: 1) computing the derivative of the regularizer in $O(D)$ operations, 2) computing the bag of visual words for each training pair by lookup in the integral histogram in $O(ND)$ operations, and 3) computing the expectation over the bag of visual words for all sides of the bounding box and for all images. We

see from Equation (5.12) that the expectation can be written as

$$\begin{aligned}
&\mathbb{E}_{y_s \sim p_{PL}(y_s | y^n_{\neg s}, x^n, w)} \left[\varphi(x^n, y^n_{\neg s}, y_s) \right] \\
&= \sum_{y_s \in \mathcal{Y}_s} \frac{\exp(\langle w, \varphi(x^n, y^n_{\neg s}, y_s) \rangle)}{Z_s(x^n, y^n_{\neg s}, w)} \varphi(x^n, y^n_{\neg s}, y_s) \\
&= \frac{1}{Z_s(x^n, y^n_{\neg s}, w)} \sum_{y_s \in \mathcal{Y}_s} \exp(\langle w, \varphi(x^n, y^n_{\neg s}, y_s) \rangle) \varphi(x^n, y^n_{\neg s}, y_s).
\end{aligned} \tag{5.16}$$

We compute the log-partition function in $O(H)$ operations and for the sum over all positions that a single side of the bounding box can take we make one lookup in the integral image in $O(1)$ operations and one lookup in the integral histogram in $O(D)$ operations. The complexity of the expectation is therefore $O(H + HD)$, and we end up with the complexity of the gradient $O(D + ND + NH + NHD)$:

$$\begin{aligned}
\nabla_w \mathcal{L}^\lambda_{PL}(w) = \underbrace{2\lambda w}_{O(D)} - \underbrace{M \sum_{n=1}^{N} \varphi(x^n, y^n)}_{O(ND)} \\
+ \underbrace{\sum_{n=1}^{N} \sum_{s=1}^{M} \mathbb{E}_{y_s \sim p_{PL}(y_s | y^n_{\neg s}, x^n, w)} \left[\varphi(x^n, y^n_{\neg s}, y_s) \right]}_{O(NH + NHD)}
\end{aligned} \tag{5.17}$$

plus the additional terms of $O(NP + NH^2)$ for setting up the integral images and $O(NP + NDH^2)$ for setting up the integral histograms.

As for the case of maximum conditional likelihood we will use LBFGS to optimize the pseudolikelihood. One iteration of LBFGS requires computing both the pseudolikelihood and the gradient at least once. We denote by K the number of iterations of LBFGS. The total complexity of the maximum pseudolikelihood problem is thus $O(K(D + N + NH + NP + NH^2 + ND + NHD + NDH^2))$ which reduces to $O(KNP + KNDH^2)$.

Compared to maximum likelihood learning, maximizing the pseudolikelihood reduces the computational complexity from $O(KNP + KNDH^4)$ to $O(KNP + KNDH^2)$.

5.3 Piecewise training

Piecewise training was introduced by Sutton and McCallum [54] who use it to approximate the likelihood function of a conditional random field. The training method approximates the maximum likelihood training by dividing the factors of the graphical model into possibly overlapping sets of pieces, each of which can be trained separately with maximum likelihood. The motivation for piecewise training is that if all individual factors fit the data well, then the resulting global distribution is likely to be reasonable.

As presented in Chapter 3 a conditional random field models the conditional distribution $p(y|x)$ and we can represent this model by a factor graph with input nodes X, output nodes Y and factors Ψ_F where F is in the set of factors $\mathcal{F}$. Each factor has a domain on the input nodes X and some subset of output nodes $Y_F \subset Y$, and we will write the value of Ψ_F with parameters w_F on some assignment (x, y) as $\Psi_F(x, y_F, w_F)$.

Let us assume that the factors of the model are divided into a set $\mathcal{P} = \{R_0, R_1, \dots\}$ of pieces where each piece $R \in \mathcal{P}$ is a set of factors $R \subset \mathcal{F}$. The pieces do not need to be disjoint. To train the pieces separately, each piece R has a local likelihood

$$p_{PW}(y_R|x, w) = \frac{1}{Z_R(x, w)} \prod_{F \in R} \Psi_F(x, y_F, w_F) \tag{5.18}$$

where the local partition function $Z_R(x, w)$ for the piece R is given by

$$Z_R(x, w) = \sum_{y_R \in \mathcal{Y}_R} \prod_{F \in R} \Psi_F(x, y_F, w_F) \tag{5.19}$$

If the pieces were disjoint and no parameters were shared between factors, then we could train each piece independently by maximizing its likelihood. In order to handle overlapping pieces and shared parameters, however, we instead train all pieces jointly by maximizing the product of the local likelihoods. For a set of pieces $\mathcal{P}$, the piecewise likelihood is

$$p_{PW}(y|x, w) = \prod_{R \in \mathcal{P}} p_{PW}(y_R|x, w) \tag{5.20}$$

$$= \prod_{R \in \mathcal{P}} \frac{1}{Z_R(x, w)} \prod_{F \in R} \Psi_F(x, y_F, w_F) \tag{5.21}$$

A choice of pieces that Sutton and McCallum [54] show a special interest in is the factor-as-piece approximation, in which each factor in the model is assigned to its own piece, that is, $\mathcal{P} = \{\{F\}|F \in \mathcal{F}\}$. We will use this choice of pieces in the approximation of the likelihood for the object localization problem.

As in the case of maximum likelihood training in Section 3.4 we assume that we are given a dataset $\mathcal{D} = \{(x^n, y^n)|n = 1, 2, \ldots, N\}$ of independent and identically distributed data pairs which has the piecewise likelihood

$$\begin{aligned} p_{PW}(y^1, \ldots, y^N | x^1, \ldots, x^N, w) &= \prod_{n=1}^{N} p_{PW}(y^n | x^n, w) \\ &= \prod_{n=1}^{N} \prod_{F \in \mathcal{F}} \frac{1}{Z_F(x^n, w_F)} \Psi_F(x^n, y_F^n, w_F). \end{aligned} \tag{5.22}$$

In the case of object localization the probabilistic graphical model is fully connected and so far we have represented it as a factor graph with a single factor Ψ and four output nodes. The factor has the form

$$\begin{aligned} \Psi(x, y, w) &= \exp\left(-E(x, y, w)\right) \\ &= \exp\left(\langle w, \varphi(x, y)\rangle\right) \end{aligned} \tag{5.23}$$

where we recall from Chapter 2 that the dot product of the weights and the bag of visual words within the region y in the image x is implemented using a precomputed integral image. Using this integral image we can compute the dot product for any region within the image with just four lookups

$$\begin{aligned} \langle w, \varphi(x, y)\rangle = \langle w, \varphi(x, y_{L,T})\rangle - \langle w, \varphi(x, y_{L,B})\rangle \\ - \langle w, \varphi(x, y_{R,T})\rangle + \langle w, \varphi(x, y_{R,B})\rangle \end{aligned} \tag{5.24}$$

where we denote by $\varphi(x, y_{a,b})$ the bag of visual words of the bounding box that has left-top corner $(0, 0)$ and right-bottom corner (a, b).

Additionally, we have the hard-constraints $y_{left} \leq y_{right}$ and $y_{top} \leq y_{bottom}$ on bounding boxes that we implicitly impose on the conditional probability $p(y|x, w)$ by only considering bounding boxes where these

constraints hold. We could, however, write the constraints explicitly in the formulation:

$$p(y|x,w) = \frac{1}{Z(x,w)} \exp(\langle w, \varphi(x,y)\rangle) \cdot \mathbb{1}_{[y_{left} \leq y_{right}]} \cdot \mathbb{1}_{[y_{top} \leq y_{bottom}]} \tag{5.25}$$

and reformulate the factor (5.23)

$$\begin{aligned} \Psi(x,y,w) &= \exp(\langle w, \varphi(x,y)\rangle) \cdot \mathbb{1}_{[y_{left} \leq y_{right}]} \cdot \mathbb{1}_{[y_{top} \leq y_{bottom}]} \\ &= \exp\left(\langle w, \varphi(x, y_{L,T})\rangle\right) \cdot \exp\left(-\langle w, \varphi(x, y_{L,B})\rangle\right) \\ &\quad \cdot \exp\left(-\langle w, \varphi(x, y_{R,T})\rangle\right) \cdot \exp\left(\langle w, \varphi(x, y_{R,B})\rangle\right) \\ &\quad \cdot \mathbb{1}_{[y_{left} \leq y_{right}]} \cdot \mathbb{1}_{[y_{top} \leq y_{bottom}]} \end{aligned} \tag{5.26}$$

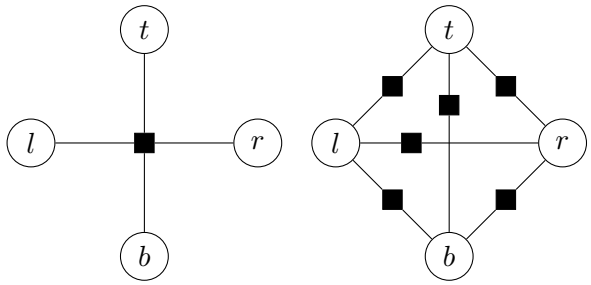
Even though the two hard-constraints do not depend on the image and the weights, for simplicity we will write all factors on the same form

$$\begin{aligned} \Psi(x,y,w) &= \Psi_{L,T}(x,y,w) \cdot \Psi_{L,B}(x,y,w) \\ &\quad \cdot \Psi_{R,T}(x,y,w) \cdot \Psi_{R,B}(x,y,w) \\ &\quad \cdot \Psi_{L,R}(x,y,w) \cdot \Psi_{T,B}(x,y,w) \\ &= \prod_{F \in \mathcal{F}} \Psi_F(x,y,w) \end{aligned} \tag{5.27}$$

with the subscript of a factor referring to the two output nodes that are within its scope: (L)eft, (T)op, (R)ight or (B)ottom.

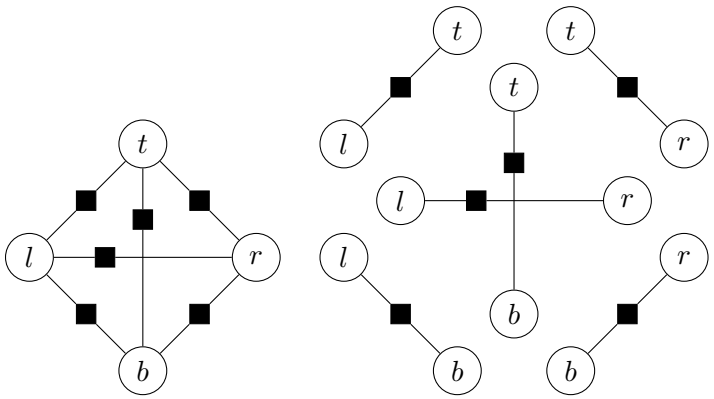
It is now clear that the fully connected factor graph can be transformed from having a single four-node-factor to one in which we have these six pairwise factors. Let us with $\mathcal{F}_\mathcal{I}$ denote the set of factors originating from the integral image computations, that is, $\mathcal{F}_\mathcal{I} = \{\{L,T\},\{R,T\},\{R,B\},\{L,B\}\}$. For simplicity of drawing let us omit drawing the observed variable x and its dependence on the factors $\Psi_F, F \in \mathcal{F}_\mathcal{I}$. See Figure 5.2 for the illustration of the factors.

We can now use piecewise training to train each of the six pairwise factors separately. The approximation to the likelihood of the original graph that piecewise training provides actually corresponds to the true likelihood in a new graph where the nodes have been split up – the so-called node-split graph. See Figure 5.3 for the node-split graph for the



(a) one four-node-factor (b) six pairwise factors

Figure 5.2 The fully connected probabilistic graphical model can be drawn with either one factor (a) or with six factors (b). For simplicity the observed variable x is not drawn.



(a) Fully connected graph (b) Node-split graph

Figure 5.3 Piecewise training approximates the likelihood of the fully connected probabilistic graphical model (a) by the likelihood of the node-split graph (b).

bounding box. Having obtained this insight, we find the optimal setting of the weights w over the dataset by maximum piecewise likelihood

$$\begin{aligned} w^*_{PW} &= \arg\max_{w \in \mathbb{R}^D} \prod_{n=1}^{N} p_{PW}(y^n | x^n, w) \\ &= \arg\min_{w \in \mathbb{R}^D} \mathcal{L}_{PW}(w) \end{aligned} \tag{5.28}$$

where the negative piecewise log-likelihood $\mathcal{L}_{PW}(w)$ is

$$\mathcal{L}_{PW}(w) = -\sum_{n=1}^{N}\sum_{F\in\mathcal{F}}\left[\log\Psi_F(x^n, y_F^n, w) - \log Z_F(x^n, w)\right], \quad (5.29)$$

and the piecewise partition function is

$$Z_F(x^n, w) = \sum_{y_F\in\mathcal{Y}_F}\Psi_F(x^n, y_F, w) \quad (5.30)$$

If we expand the double sum in (5.29),

$$\begin{aligned}
&\sum_{n=1}^{N}\sum_{F\in\mathcal{F}}\left[\log\Psi_F(x^n, y_F^n, w) - \log Z_F(x^n, w)\right] \\
&= \sum_{n=1}^{N}\sum_{F\in\mathcal{F}_\mathcal{I}}\left[\log\Psi_F(x^n, y_F^n, w) - \log Z_F(x^n, w)\right] \\
&\quad + \sum_{n=1}^{N}\left[\log\mathbb{1}_{[y_{left}\le y_{right}]} - \log\sum_{\substack{y_{left}\in\mathcal{Y}_{left}\\ y_{right}\in\mathcal{Y}_{right}}}\mathbb{1}_{[y_{left}\le y_{right}]}\right] \\
&\quad + \sum_{n=1}^{N}\left[\log\mathbb{1}_{[y_{top}\le y_{bottom}]} - \log\sum_{\substack{y_{top}\in\mathcal{Y}_{top}\\ y_{bottom}\in\mathcal{Y}_{bottom}}}\mathbb{1}_{[y_{top}\le y_{bottom}]}\right],
\end{aligned} \quad (5.31)$$

we are reminded that the factor terms corresponding to the hard-constraints $y_{left} \le y_{right}$ and $y_{top} \le y_{bottom}$ do not depend on the weights w and so they will just be a constant in the objective function

$$\begin{aligned}
\mathcal{L}_{PW}(w) &= -\sum_{n=1}^{N}\sum_{F\in\mathcal{F}_\mathcal{I}}\left[\log\Psi_F(x^n, y_F^n, w) - \log Z_F(x^n, w)\right] + Const \\
&= -\sum_{n=1}^{N}\sum_{F\in\mathcal{F}_\mathcal{I}}\left[\text{sign}_F\cdot\langle w, \varphi(x^n, y_F^n)\rangle - \log Z_F(x^n, w)\right] + Const
\end{aligned} \quad (5.32)$$

where sign_F multiplies by -1 if the term in the computation of the dot product in Equation (5.24) corresponding to the factor is negative.

If we continue computing on the factor terms corresponding to the hard-constraints we get

$$\begin{aligned}
&\sum_{n=1}^{N}\left[\log \mathbb{1}_{\left[y_{left}\leq y_{right}\right]} - \log \sum_{\substack{y_{left}\in\mathcal{Y}_{left}\\ y_{right}\in\mathcal{Y}_{right}}} \mathbb{1}_{\left[y_{left}\leq y_{right}\right]}\right] \\
&\quad + \sum_{n=1}^{N}\left[\log \mathbb{1}_{[y_{top}\leq y_{bottom}]} - \log \sum_{\substack{y_{top}\in\mathcal{Y}_{top}\\ y_{bottom}\in\mathcal{Y}_{bottom}}} \mathbb{1}_{[y_{top}\leq y_{bottom}]}\right] \\
&= -\sum_{n=1}^{N}\left[\log \sum_{\substack{y_{left}\in\mathcal{Y}_{left}\\ y_{right}\in\mathcal{Y}_{right}}} \mathbb{1}_{\left[y_{left}\leq y_{right}\right]} + \log \sum_{\substack{y_{top}\in\mathcal{Y}_{top}\\ y_{bottom}\in\mathcal{Y}_{bottom}}} \mathbb{1}_{[y_{top}\leq y_{bottom}]}\right] \\
&= -\sum_{n=1}^{N}\log \sum_{y\in\mathcal{Y}} \mathbb{1}_{\left[y_{left}\leq y_{right}\wedge y_{top}\leq y_{bottom}\right]}.
\end{aligned} \tag{5.33}$$

We see that they are actually a constant related to the count of legal bounding over all training images. Since we are optimizing, we do not care about what that value is.

As in the case of the maximum conditional likelihood in Section 3.4, minimizing the negative piecewise log-likelihood in Equation (5.32) will be prone to overfitting, and so we will add an L_2-regularizer and instead minimize the regularized negative piecewise log-likelihood

$$\begin{aligned}
\mathcal{L}^{\lambda}_{PW}(w) = \lambda||w||^2 - \sum_{n=1}^{N}\sum_{F\in\mathcal{F}_{\mathcal{I}}}\Big[&\text{sign}_F \cdot \langle w, \varphi_F(x^n, y^n_F)\rangle \\
&- \log Z_F(x^n, w)\Big] + Const
\end{aligned} \tag{5.34}$$

The gradient of the negative piecewise log-likelihood with respect

to the weights is

$$\begin{aligned}\nabla_w \mathcal{L}_{PW}(w) &= \frac{\partial}{\partial w}\left(-\sum_{n=1}^{N} \sum_{F \in \mathcal{F}_\mathcal{I}} \left[\text{sign}_F \cdot \langle w, \varphi(x^n, y_F^n) \rangle - \log Z_F(x^n, w) \right] \right) \\ &= -\sum_{n=1}^{N} \sum_{F \in \mathcal{F}_\mathcal{I}} \left[\text{sign}_F \cdot \varphi(x^n, y_F^n) - \frac{\partial}{\partial w} \log Z_F(x^n, w) \right].\end{aligned} \tag{5.35}$$

The gradient of the log-partition function with respect to the weights is

$$\begin{aligned}\frac{\partial}{\partial w} \log Z_F(x^n, w) &= \frac{1}{Z_F(x^n, w)} \frac{\partial}{\partial w} Z_F(x^n, w) \\ &= \text{sign}_F \cdot \sum_{y_F \in \mathcal{Y}_F} \frac{\exp(\text{sign}_F \cdot \langle w, \varphi(x^n, y_F) \rangle)}{Z_F(x^n, w)} \varphi(x^n, y_F) \\ &= \text{sign}_F \cdot \sum_{y_F \in \mathcal{Y}_F} p_{PW}(y_F | x^n, w) \varphi(x^n, y_F) \\ &= \text{sign}_F \cdot \mathbb{E}_{y_F \sim p_{PW}(y_F | x^n, w)} \left[\varphi(x^n, y_F) \right],\end{aligned} \tag{5.36}$$

and so we have

$$\begin{aligned}\nabla_w \mathcal{L}_{PW}(w) = -\sum_{n=1}^{N} \sum_{F \in \mathcal{F}_\mathcal{I}} \Bigg[&\text{sign}_F \cdot \varphi(x^n, y_F^n) \\ &- \text{sign}_F \cdot \mathbb{E}_{y_F \sim p_{PW}(y_F | x^n, w)} \left[\varphi(x^n, y_F) \right] \Bigg].\end{aligned} \tag{5.37}$$

The gradient of the regularized negative piecewise log-likelihood is likewise

$$\begin{aligned}\nabla_w \mathcal{L}^\lambda_{PW}(w) = 2\lambda w - \sum_{n=1}^{N} \sum_{F \in \mathcal{F}_\mathcal{I}} \Bigg[&\text{sign}_F \cdot \varphi(x^n, y_F^n) \\ &- \text{sign}_F \cdot \mathbb{E}_{y_F \sim p_{PW}(y_F | x^n, w)} \left[\varphi(x^n, y_F) \right] \Bigg].\end{aligned} \tag{5.38}$$

5.3.1 Computational complexity

We will now derive the complexity of maximizing the piecewise likelihood. As in Section 3.6, we denote by N the number of training examples, P the number of feature points in an image, H the height of an image, and D the dimensionality of w.

Computing the regularized negative piecewise log-likelihood (5.34) consists of first computing the regularizer in $O(D)$ operations and then for each image we must: 1) compute the dot product of w with the bag of words in $O(1)$ operations using the integral image and 2) compute the piecewise log-partition function. The partition function does a lookup in constant time summing over two sides of the bounding box in $O(H^2)$ operations. This gives a complexity of $O(D + N + NH^2)$:

$$\begin{aligned}\mathcal{L}^{\lambda}_{PW}(w) = \underbrace{\lambda||w||^2}_{O(D)} - \sum_{n=1}^{N} \sum_{F \in \mathcal{F}_{\mathcal{I}}} \Bigg[\underbrace{\text{sign}_F \cdot \langle w, \varphi(x^n, y^n_F) \rangle}_{O(1)} \\ - \underbrace{\log Z_F(x^n, w)}_{O(H^2)} \Bigg]\end{aligned} \tag{5.39}$$

plus the additional term of $O(NP + NH^2)$ for setting up the integral images.

Computing the derivative of the regularized negative piecewise log-likelihood (5.38) consists of computing the gradient of the regularizer in $O(D)$ operations, then for each training pair we must: 1) compute the bag of visual words by a lookup in the integral histogram in $O(D)$ operations and 2) compute the expectation over the bag of visual words for two sides of the bounding box. We see from Equation (5.36) that the expectation can be written

$$\begin{aligned}&\mathbb{E}_{y_F \sim p_{PW}(y_F|x^n,w)} \left[\varphi(x^n, y_F)\right] \\ &= \sum_{y_F \in \mathcal{Y}_F} \frac{\exp(\langle w, \varphi(x^n, y_F) \rangle)}{Z_F(x^n, w)} \varphi(x^n, y_F) \\ &= \frac{1}{Z_F(x^n, w)} \sum_{y_F \in \mathcal{Y}_F} \exp(\langle w, \varphi(x^n, y_F) \rangle) \varphi(x^n, y_F)\end{aligned} \tag{5.40}$$

and we compute the log-partition function in $O(H^2)$ operations and for the sum over all positions that two sides of the bounding box can take we make one lookup in the integral image in $O(1)$ operations and one lookup in the integral histogram in $O(D)$ operations. The complexity of the expectation is therefore $O(H^2 + H^2D)$, and we end up with the complexity of the gradient $O(D + ND + NH^2 + NH^2D)$:

$$\nabla_w \mathcal{L}_{PW}^{\lambda}(w) = \underbrace{2\lambda w}_{O(D)} - \sum_{n=1}^{N} \sum_{F \in \mathcal{F}_{\mathcal{I}}} \Bigg[\underbrace{\text{sign}_F \cdot \varphi(x^n, y_F^n)}_{O(D)} - \underbrace{\text{sign}_F \cdot \mathbb{E}_{y_F \sim p_{PW}(y_F|x^n,w)} \left[\varphi(x^n, y_F)\right]}_{O(H^2+H^2D)} \Bigg] \tag{5.41}$$

plus the additional terms of $O(NP + NH^2)$ for setting up the integral images and $O(NP + NDH^2)$ for setting up the integral histograms.

We will use LBFGS to optimize the piecewise likelihood and let K be the number of iterations for LBFGS to converge. The total complexity of the maximum piecewise likelihood problem is thus $O(K(D + N + NH^2 + NP + ND + NP + NDH^2))$ which reduces to $O(KNP + KNDH^2)$. Compared to maximum likelihood learning, using piecewise training reduces the computational complexity from $O(KNP + KNDH^4)$ to $O(KNP + KNDH^2)$.

5.4 Summary

We studied two approximations of the likelihood at training time, namely the pseudolikelihood and the piecewise likelihood. Pseudolikelihood approximates by conditioning each output node in the graph on its neighbors and then maximizing these node-conditional probabilities independently. Piecewise training approximates by dividing the factors of the graphical model into possibly overlapping sets of pieces, and then trains the factors independently with maximum likelihood. For the problem of object localization both approximations reduce the computational complexity of maximum likelihood learning from $O(KNP + KNDH^4)$ to $O(KNP + KNDH^2)$.

6

Experiments

In this chapter we present a comparison between the exact maximum conditional likelihood and the four approximation methods, stochastic gradient descent, contrastive divergence, pseudolikelihood and piecewise training. Furthermore, we show that contrastive divergence, pseudolikelihood and piecewise training produce localization results comparable with the results obtained by max-margin learning.

6.1 Introduction

We now investigate the four approximate learning algorithms that we presented in Chapter 4 and 5. First we will compare these methods against the exact maximum conditional likelihood (MCL) on a dataset where full log-likelihood optimization is tractable with suitable parallelization. As explained in Chapter 2 we measure performance by the area under the overlap-precision curve (AUC). We will compare AUC scores and time consumption of all methods and show that only contrastive divergence, pseudolikelihood and piecewise training are tractable.

These three tractable approximation schemes will then be used to

perform localization in the more complex object detection problem of PASCAL VOC 2006 dataset. We compare our results with the ones obtained by Blaschko and Lampert using a structured support vector machine (SVM) [6]. See Appendix A for a short description of the structured SVM.

6.2 Experimental setup

We designed and implemented in C++ the CRF model and all the objective functions and gradients as well as the approximate learning algorithms. For LBFGS we used the libLBFGS library[1] by Naoaki Okazaki. For prediction we used the Efficient Subwindow Search (ESS) algorithm by Lampert [38]. The U-SURF features for both datasets were kindly provided by Lampert. Model selection, training and testing were performed on a 320 core HPC cluster (20 nodes, each with 64 GB memory).

When running the LBFGS procedure we use the norm of the gradient as well as a relative difference in objective value as stop criteria. For the stochastic gradient methods the stop criterion is a predefined maximum number of epochs. We choose 50 epochs for SGD and 200 epochs for CD. Furthermore, to accelerate the computation of the objective and the gradient we quantize the coordinates of each bounding box with a step size of 8 pixels.

6.3 Results on TU Darmstadt cow

The TU Darmstadt `cow` images are relatively small and each image shows exactly one cow with different backgrounds. Since the images are small, computing both the log-likelihood and the gradient is tractable and we can perform parameter learning with all five methods on this dataset. Some example images from the dataset are shown in Figure 6.1. The official dataset consists of 111 training images and 557 test images. However, in order to perform model selection we selected 20 representative images from the training set, leaving 91 images for validation. Since all the images are very similar with one cow in the middle of the

[1] `http://www.chokkan.org/software/liblbfgs/`

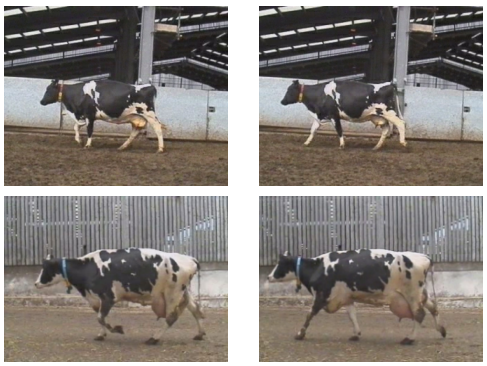


Figure 6.1 Examples from the TU Darmstadt cow dataset.

image, we estimated that 20 images be enough to evaluate the performance during model selection. Once a suitable λ has been found, we retrain on the whole training set. See Appendix B for details.

The training and validation AUC scores for different values of λ are listed in Figure 6.2, and AUC scores for the test set are shown in Table 6.1. Also shown is the selected λ, the number of passes through the dataset, the overall training time and the time per pass. To accelerate the MCL training, the LBFGS method has been parallelized to use 56 cores such that each core processes at most two images when computing the objective and the gradient. Note that the reason for SGD being much slower than MCL is that SGD was not parallelized and was run for more passes through the dataset than MCL.

The precision-overlap curves for the test set are depicted in Figure 6.3. The curves are almost identical which indicates that all training methods have learned weights that model the bounding box distribution equally well. This indicates that the approximate learning methods are actually good candidates for replacing exact parameter learning.

6.4 Results on PASCAL VOC 2006

The PASCAL VOC 2006 dataset consists of images of ten different object categories. In each image there can be several different objects as well as several objects of the same category. In addition, each object can be occluded by another object or only partially shown. This

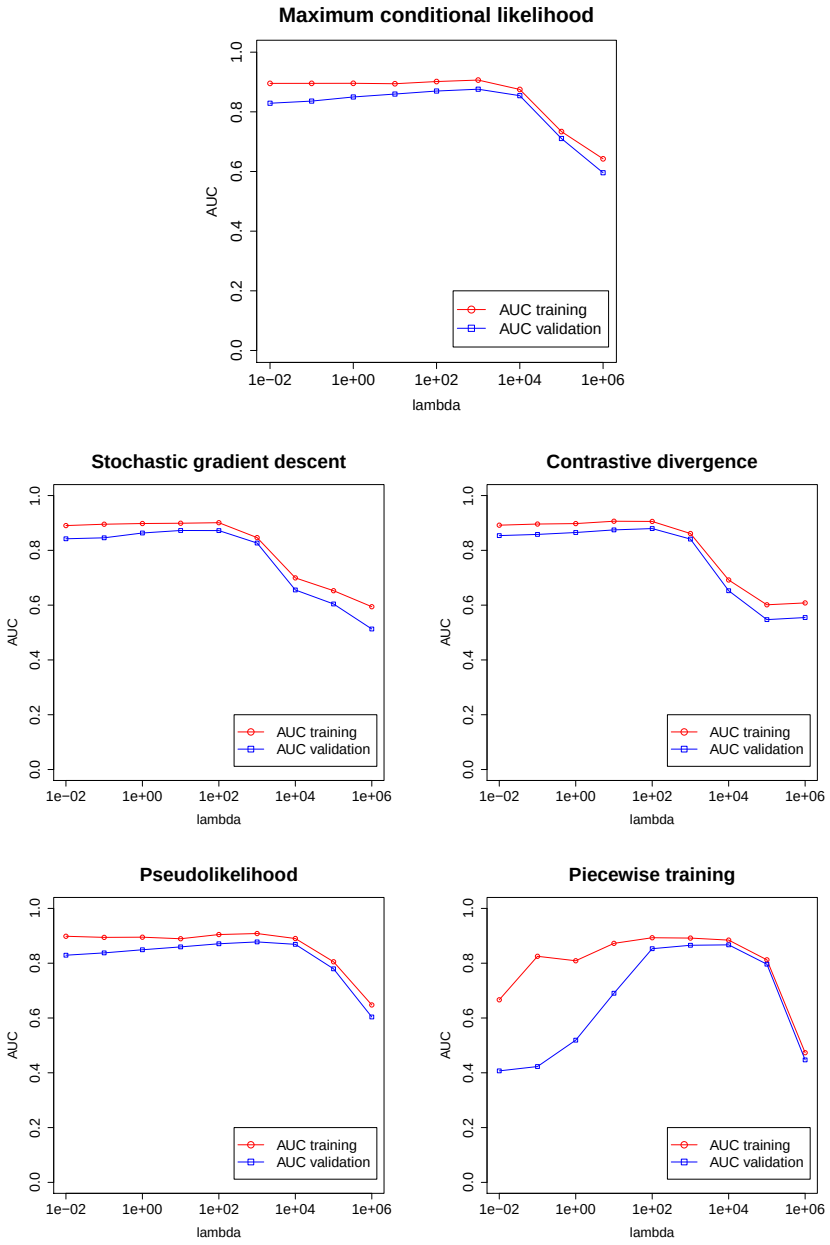


Figure 6.2 Model selection curves for the TU Darmstadt `cow` dataset. Piecewise training is especially sensitive to the regularization parameter.

TU Darmstadt cow dataset					
method	λ	passes	time (s)	time/pass*	AUC test
MCL	1000	26	14062.9	30289.32	0.848754
SGD	10	50	897799	17956	0.848402
CD	100	200	10236.7	51.18	0.835903
PL	1000	33	1227.45	37.20	0.851402
PW	10000	50	2498.56	49.97	0.810254

Table 6.1 Method comparison. *As LBFGS was parallelized to 56 cores the time/pass is computed as if one core was used.

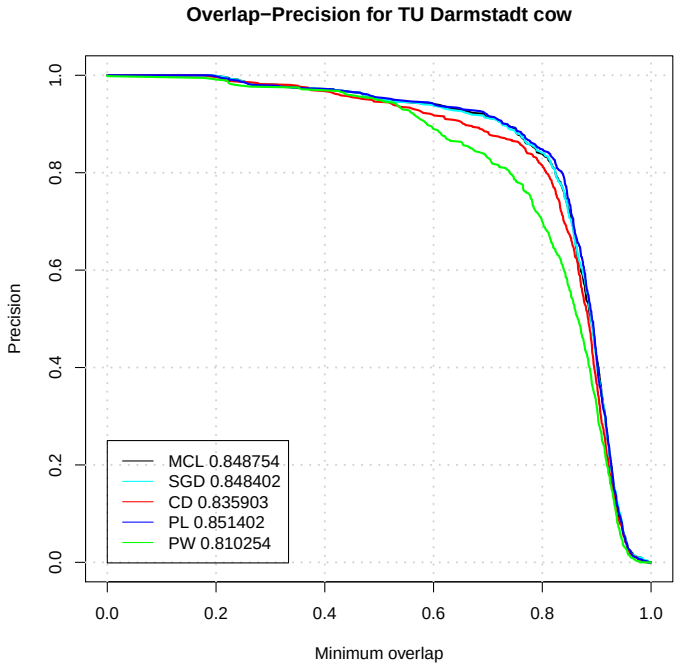


Figure 6.3 Overlap-precision curves for TU Darmstadt cow test set.

makes the PASCAL dataset especially challenging. See some examples in Figure 6.4.

As the complexity analyses in the previous chapters suggest and as we experimentally verified on the TU Darmstadt dataset, MCL and

Figure 6.4 Examples from the PASCAL VOC 2006 dataset.

SGD are too slow for the purpose of training the CRF model. Furthermore, the results obtained by all approximate methods are very similar to the results obtained by exact training. Therefore, we will only investigate the performance of CD, PL and PW for the PASCAL VOC 2006 dataset. The dataset is already divided into a training set, a validation set and a test set (see Appendix B for details). We create a CRF model for each of the ten object categories.

The regularization parameter λ is chosen by performing model selection on the training and validation set. We trained 12 models for each object category with $\lambda = \{10^{-5}, 10^{-4}, \ldots, 10^{5}, 10^{6}\}$ and for each object category we chose the λ that performed best on the validation set. In Figure 6.5 we see the training and validation performance for different values of λ for all three learning methods on the bicycle images. For all three methods we see that the validation AUC peaks for a specific value of λ before decreasing again. Similarly, we see that the training AUC is almost constant for small values of λ and then starts decreasing for large values. This shows that the model overfits with too little regularization and underfits with too much regularization.

We selected the λ value based on the performance on the validation set and then computed overlap-precision on the test set. In Table 6.2

CD on PASCAL VOC 2006			
object	λ	val	test
bicycle	100	0.52912	0.514949
bus	10	0.478664	0.49383
car	10	0.505873	0.453237
cat	100	0.560591	0.540866
cow	100	0.5826	0.532994
dog	10	0.502776	0.480479
horse	100	0.497044	0.481811
motorbike	10	0.508116	0.560153
person	100	0.277626	0.293902
sheep	100	0.450955	0.450475

Table 6.2 Contrastive divergence overlap-precision: Validation and test performance for the best values of λ.

we see the results obtained on the validation and test sets for the best value of the regularization parameter using CD. We see that the AUC is slightly better on the validation set than the test set for most categories. We also notice that the category `person` is especially hard to localize and achieves much smaller AUC than the other objects. Results of the same evaluation setup for PL and PW are listed in Table 6.3 and Table 6.4 where the same observations are made. These tests were all performed with a quantization step size of 8 but very similar results were obtained with a step size of 4.

To achieve the best possible performance we retrain each model on the combined training and validation set with the optimal λ just found. This also makes us able to compare with the results obtained by Blaschko and Lampert [6] using a structured SVM. The test results obtained after having trained on the combined training-validation set is shown in Table 6.5. Also shown are the results obtained by the structured SVM as well as results obtained by using random weights in the interval $[-0.0001, 0.0001]$. As could be expected, the random weights give much worse results than the trained models which indicates that these models have actually learned something valuable from the training.

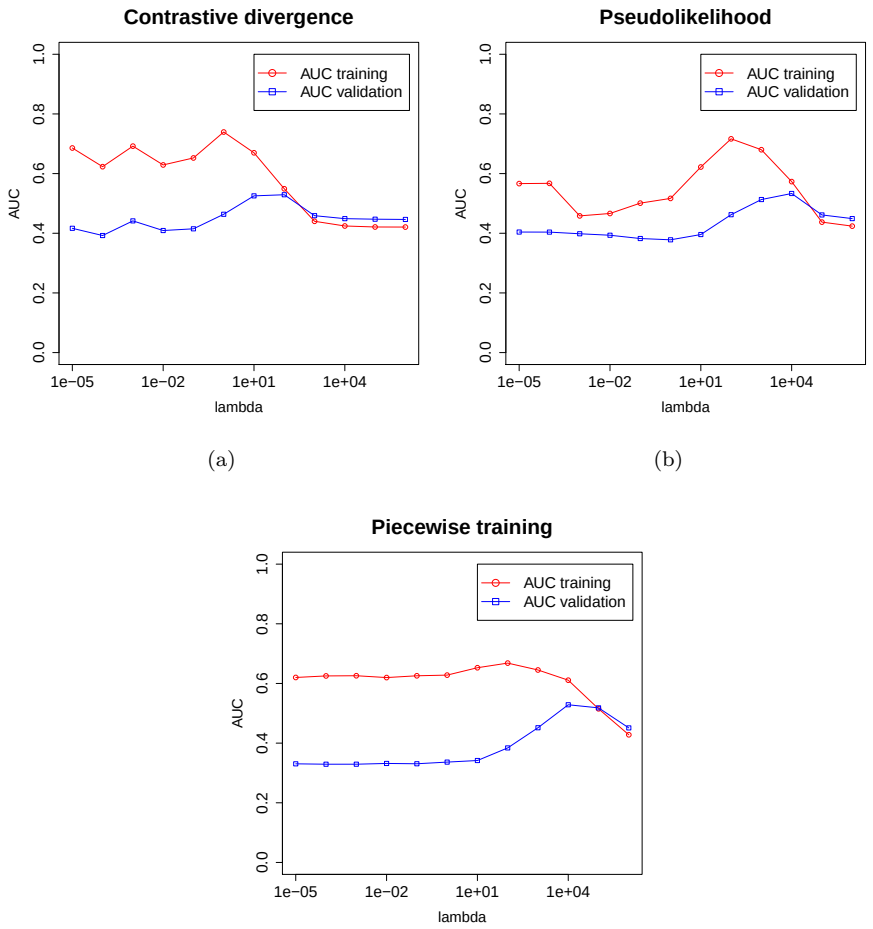


Figure 6.5 The training and validation AUC for PASCAL bicycle for all three methods: (a) contrastive divergence, (b) pseudolikelihood and (c) piecewise training.

The best AUC is marked in the table and there is no indication that one method is better than any other. A Wilcoxon Signed-Rank test confirmed that there is no significant difference between the results. From this we conclude that contrastive divergence, pseudolikelihood and piecewise training perform comparably with structured SVM training.

PL on PASCAL VOC 2006			
object	λ	val	test
bicycle	10000	0.533283	0.52125
bus	1000	0.468187	0.488581
car	10000	0.492438	0.446909
cat	10000	0.571087	0.546109
cow	10000	0.576236	0.527904
dog	10000	0.499569	0.491968
horse	10000	0.486708	0.480639
motorbike	10000	0.520168	0.565627
person	10000	0.267498	0.265702
sheep	10000	0.444999	0.439965

Table 6.3 Pseudolikelihood overlap-precision: Validation and test performance for the best values of λ.

PW on PASCAL VOC 2006			
object	λ	val	test
bicycle	10000	0.528589	0.531953
bus	10000	0.492332	0.503328
car	10000	0.525323	0.470283
cat	100000	0.572708	0.546801
cow	10000	0.568462	0.524374
dog	10000	0.499105	0.473258
horse	10000	0.475837	0.482592
motorbike	100000	0.4978	0.544059
person	100000	0.273866	0.288465
sheep	100000	0.476964	0.446147

Table 6.4 Piecewise overlap-precision: Validation and test performance for the best values of λ.

The overlap-precision curves for `motorbike` and `person` are shown in Figure 6.6. We see that for `motorbike`, where we achieved an AUC around 0.57 for all three methods, a minimum overlap of 0.5 gives a precision of approximately 0.65, meaning that 65% of all predictions have an overlap above 50% with a ground truth box. For `person` the

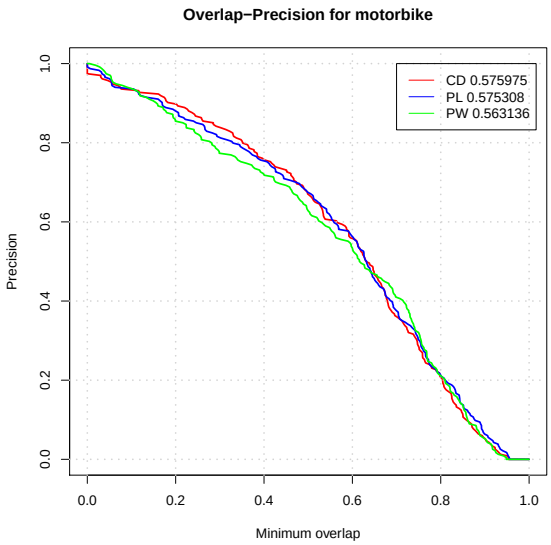
Comparison with SSVM and random weights					
object	CD	PL	PW	SSVM	random
bicycle	0.523872	0.536738	0.52625	**0.548046**	0.425627
bus	0.508536	0.484125	**0.516445**	0.512325	0.334877
car	0.45572	0.463441	**0.479368**	0.475305	0.284984
cat	0.529061	0.549859	0.555661	**0.557535**	0.394224
cow	**0.531107**	0.524136	0.517867	0.531008	0.305451
dog	0.497916	**0.501726**	0.474153	0.375558	0.34579
horse	0.489276	0.489673	0.48282	**0.49625**	0.310298
motorbike	0.576262	0.575207	0.562999	**0.577624**	0.366089
person	**0.301073**	0.286617	0.296874	0.293937	0.181158
sheep	0.456576	0.455665	**0.466451**	0.436382	0.243538

Table 6.5 Test errors for contrastive divergence, pseudolikelihood, piecewise log-likelihood, structured SVM and random weights in the interval $[-0.0001, 0.0001]$. Training performed on the `trainval` set.

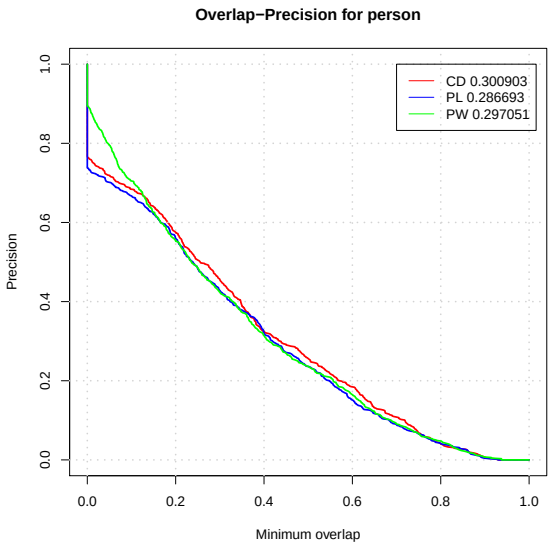
precision is around 0.25 and even for small overlaps, the precision is less than 0.8 for CD and PL.

6.5 Summary

We presented our experimental setup and compared MCL learning with the four approximate methods. Results on a tractable dataset showed that the approximations obtained results comparable with MCL. Furthermore, we investigated the performance of contrastive divergence, pseudolikelihood and piecewise training on the PASCAL VOC 2006 dataset and showed that they obtain results comparable with structured SVM learning.



(a) motorbike



(b) person

Figure 6.6 Overlap-precision curves for motorbike and person.

7

Discussion

In this chapter we discuss the results presented in the previous chapter and analyze the bounding box distribution represented by the trained CRF model. Furthermore, we discuss future work and suggest extensions of the model.

7.1 Introduction

The results obtained by training the CRF model with the different learning algorithms turn out to yield almost identical results on both datasets. This is certainly an argument in favor of using the fastest training method available. However, the number of epochs for SGD and CD were chosen ad hoc and more careful selection of stopping criteria could accelerate these methods.

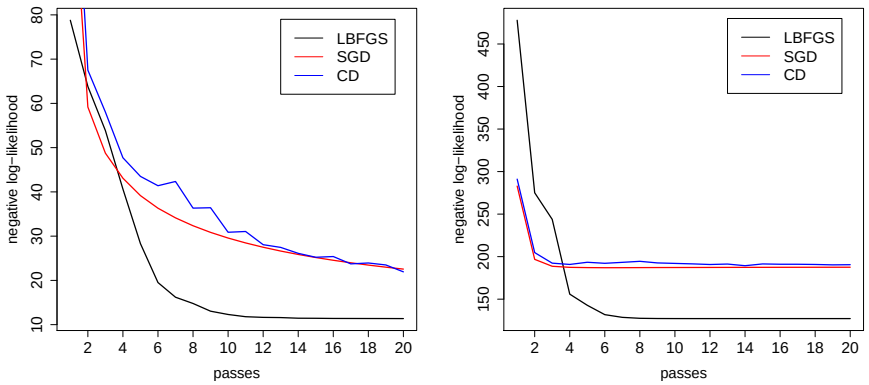
Pseudolikelihood and piecewise training optimized a different objective with special assumptions. We will see that these assumptions make sense for the localization task by visualizing the corner marginals. Also, the Gibbs sampler used in contrastive divergence gives us the possibility of sampling bounding boxes from the learned distribution. This can be used to show how peaked the distribution is around objects.

7.2 Stochastic gradient approximations

The stochastic gradient methods optimize the real objective function, the negative log-likelihood, so we can compare the weights in terms of objective value with the ones obtained by the LBFGS procedure. In Figure 7.1 we see the values of the negative log-likelihood for LBFGS, SGD and CD for 20 passes of the TU Darmstadt training set, that is, 20 iterations of LBFGS and 20 epochs of SGD and CD. The plot on the left shows the objective for $\lambda = 0.01$ and the plot on the right shows the objective for $\lambda = 1000$. There are a few interesting things to note here: First, both stochastic gradient methods are unable to reach the same objective value as the LBFGS method. This is a consequence of the approximate gradient and the choice of learning rate. Secondly, the regularization parameter influences the convergence rate of the methods. For small values of λ convergence takes longer for all three methods, but effect seems stronger for the stochastic gradient methods. For $\lambda = 0.01$ neither SGD nor CD seem to have converged after 20 passes of the dataset whereas LBFGS seems to have converged after 14 iterations. For $\lambda = 1000$ the situation is reversed and both SGD and CD converge after only 3 epochs and LBFGS converges after 8 iterations.

As shown in Chapter 4 contrastive divergence learning is similar to SGD with the only difference being that the real expectation is replaced by a single sample obtained by running one step in the Gibbs chain. This approximation of the expectation seems to have CD achieve slightly worse objective value than SGD. Also, the CD gradient is more stochastic, which results in the jumps of the objective seen in the left plot. The SGD is more consistently decreasing the objective after each epoch, although jumps occur between iterations as we saw in Chapter 4.

Although the stochastic gradient methods do not achieve an objective close to the minimum we saw that the test performance is actually very similar to the one obtained by the LBFGS. This shows that for learning it is worth sacrificing optimality in the objective for efficiency. Bottou and Bousquet [9] discuss this issue in more detail for the case of large datasets.



(a) (b)

Figure 7.1 Values of the negative log-likelihood during 20 passes through the training set with $\lambda = 0.01$ (a) and $\lambda = 1000$ (b). The stochastic gradient methods converge quickly in the first couple of epochs but then slow down drastically.

7.3 Log-likelihood approximations

Compared to the stochastic gradient approximations the log-likelihood approximations optimize alternative objective functions exactly. In pseudolikelihood we model the joint distribution as a product of conditioned distributions and in piecewise training we model the joint distribution as a product of piecewise distributions. It is not obvious that optimizing these alternative objectives leads to weight vectors that parameterize the bounding box distribution well. However, the results indicate that they do work very well in practice.

To see beyond numeric results why a set of learned weights works well, we visualize the four-dimensional distribution by computing each of the four corner marginals of the CRF model and plot these against the ground truth bounding box. In Figure 7.2(a) we see the corner marginals of the CRF model with the weights obtained by PL on a test image. Each of the four corner marginals is represented by an individual color. We also visualize the assumptions made my PL by computing what we will denote as *corner pseudo-marginals*. These are given by a product of two conditional coordinate distributions given

the ground truth bounding box. As an example, the left-top corner pseudo-marginal we compute by

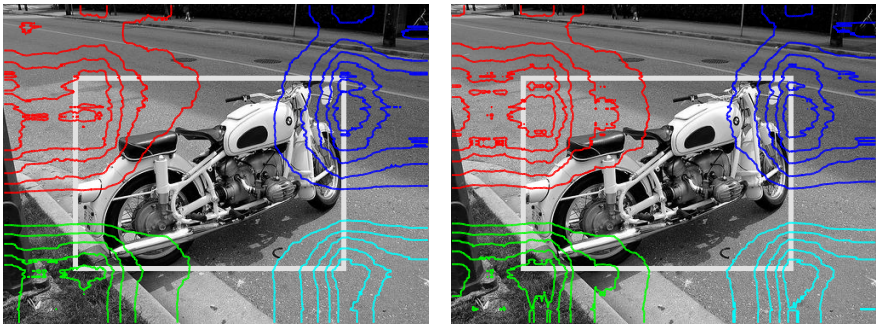
$$p_{pseudo}(y_L, y_T) = p_{PL}(y_L|y_T, y_R, y_B, x, w) \cdot p_{PL}(y_T|y_L, y_R, y_B, x, w). \tag{7.1}$$

In Figure 7.2(b) we see the corner pseudo-marginals with the weights obtained by PL on a test image. We see that both the corner marginals in (a) and the corner pseudo-marginals in (b) are peaked around the corners of the object. This indicates that the PL assumptions are reasonable to make during training.

Similarly, in Figure 7.3 (a) and (b) we see the corner marginals of the CRF distribution with the weights obtained by piecewise training and the corner marginals of the piecewise distribution. For the CRF distribution the corner marginals are likewise located around the corners of the object but are less peaked. The marginal distributions seem to be identical which again indicates that the piecewise training makes sense. The motivation behind piecewise training is that it makes sense if the factors individually model the distribution well. In Figure 7.3 (c) we plot the individual factor distributions and we see an interesting phenomenon: the factor distributions are pairwise identical. With the assumption that each factor is independent of the others, the left-top factor and the right-bottom factor try to determine a region with origin $(0, 0)$ that includes as much of the motorbike and as little background as possible. The right-top and left-bottom factors try to determine a region that includes as many non-motorbike features and as few motorbike features as possible. Therefore, we see that the the left-top and right-bottom factors have maximum around the bottom-right corner of the true bounding box and that the right-top and left-bottom factors have their maximum around the middle of the motorbike. This indicates that the factors individually model the distribution well.

7.4 Sampling

The Gibbs sampler can be used to yield further insight into the bounding box distribution. In Figure 7.4 we see some example images where we have sampled bounding boxes starting from random configurations and let the chain run for 1000 steps. For the TU Darmstadt cows,



(a) Corner marginals for CRF with pseudo-likelihood weights.

(b) Corner marginals for pseudolikelihood distribution.

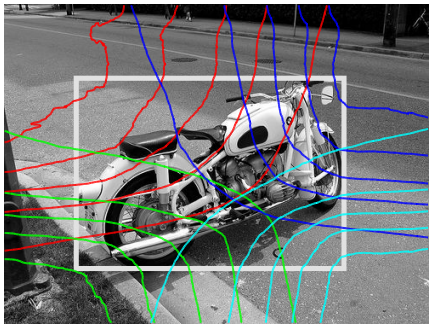
Figure 7.2 Corner marginals for pseudolikelihood. Red is top-left marginal, blue is top-right marginal, green is bottom-left marginal and cyan is bottom-right marginal.

where we achieved high localization performance on the test set, we see that many of the sampled boxes end up including parts of the cow. This indicates that the distribution indeed has a mode around the cow. However, for the PASCAL VOC 2006 dataset the distribution is not as peaked.

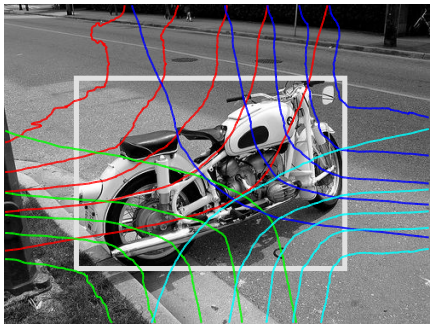
When more than one instance of the object is present in the image, the distribution tends to favor samples that cover all objects instead of single objects. Since we do not give a lower score to boxes that include more than one object instance, it makes sense that the distribution favors boxes that include more object features. This can also be seen from the heatmap shown in Figure 7.4. This is a visual representation of the importance of each feature point where red color means high weights and blue color means low weights. Including more objects without including too much more background will increase the overall score of the box.

7.5 Future work

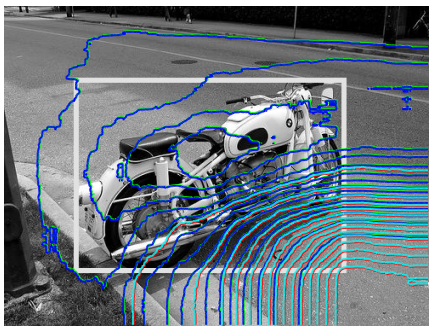
We have introduced a new model for object localization and investigated four approximate algorithms for parameter learning. In the following we give some suggestions for future directions.



(a) Corner marginals for CRF with piecewise training weights.



(b) Corner marginals for piecewise distribution.



(c) Factor marginals for piecewise distribution.

Figure 7.3 Corner marginals and factor distributions for piecewise training. For (a) and (b), red is left-top marginal, blue is right-top marginal, green is left-bottom marginal and cyan is right-bottom marginal. In (c) the same colors illustrate the factor distributions.

7.5.1 Stochastic gradient parameters

The stochastic gradient approximations yielded a whole family of methods, parameterized by the number of images per batch, the number of samples and the number of Gibbs chain steps. This family needs to be investigated for optimal tradeoff between convergence rate and computation time. Using more samples and running the Gibbs chain for more steps will give more accurate gradients but the question is whether this will increase the convergence rate. In Figure 7.1 we saw that CD gradient (using one sample and one Gibbs step) achieved convergence

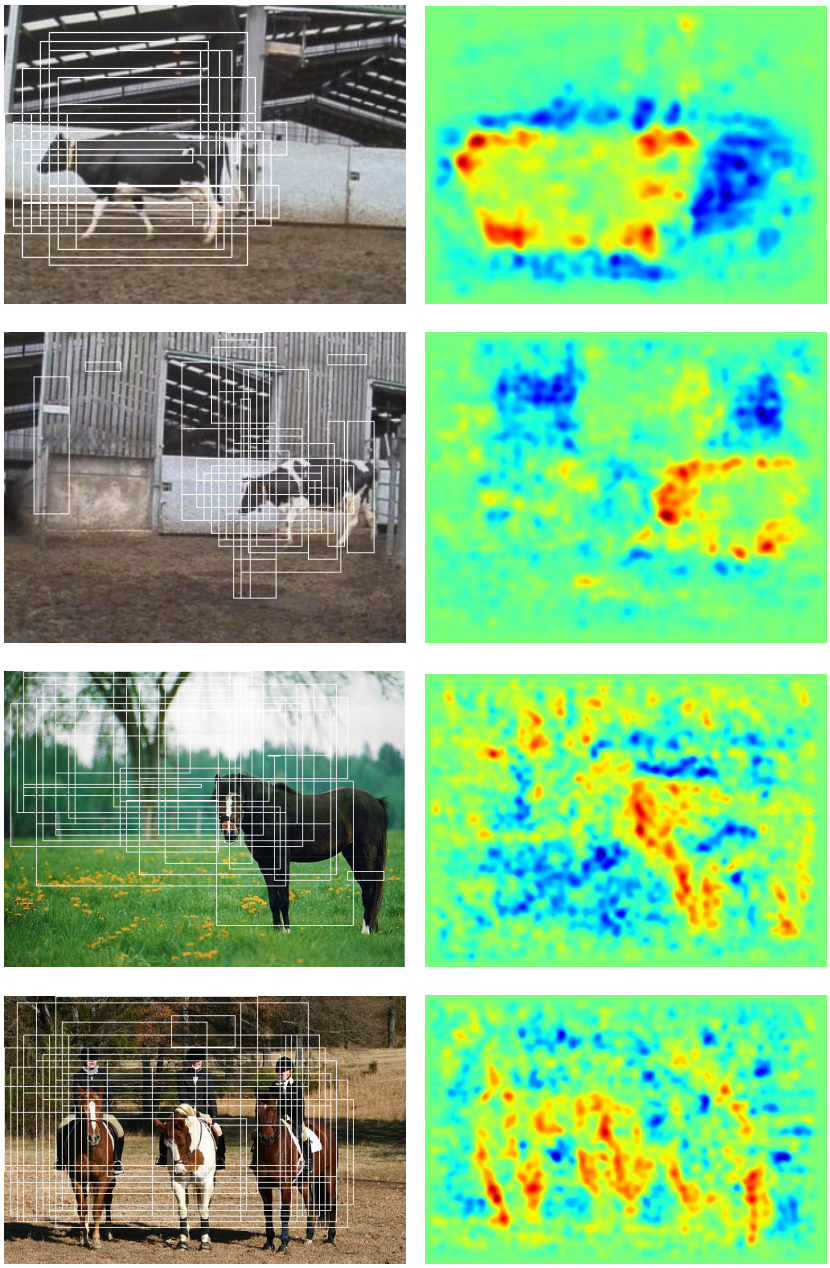


Figure 7.4 Samples obtained by the Gibbs sampler (left column) and heatmaps that indicates the importance of the features for the given object category (right column).

rates close to the SGD which computed the true expectation for each image. This suggests that keeping the number of samples and steps low is a good tradeoff. However, it could be interesting to investigate the methods using a small batch of images instead of just a single image.

7.5.2 Other learning algorithms

Other approximate learning algorithms could be considered. Kumar et al. [34] suggest an alternative approximation of the gradient inspired by contrastive divergence, where the expectation is replaced by the highest scoring bounding box instead of a sampled bounding box. This is called saddle point approximation (SPA). If we assume one object instance per image, this corresponds to minimizing the distance between the highest scoring box and the ground truth.

Piecewise pseudolikelihood suggested by Sutton and McCallum [53] accelerates piecewise training by performing maximum pseudolikelihood instead of exact maximum likelihood in the node-split graph.

7.5.3 Multiple object instances

The proposed model trains the weights assuming only one object per image. To detect two objects in an image it may seem logical to retrieve the two highest scoring bounding boxes obtained by the prediction. However, it is very likely that the second highest scoring box is placed very close to the highest scoring box, covering the same object. A technique called *non-maximum suppression* can be used to filter out boxes that lie too close to the highest scoring box and thereby enable prediction of boxes around other object instances.

A more principled approach would be to create a CRF model that explicitly models the distribution of several boxes. This could be done by extending the CRF model in Figure 3.5 to include a variable indicating how many objects are present in the image and then "activate" the corresponding bounding box variables. For example, in Figure 7.5 we see a CRF model for detecting two bounding boxes where the random variable n indicates how many objects are present in the image. If one object is present, only the first bounding box variables (l_1, t_1, r_1, b_1) are activated, which can be implemented by setting (l_2, t_2, r_2, b_2) to some

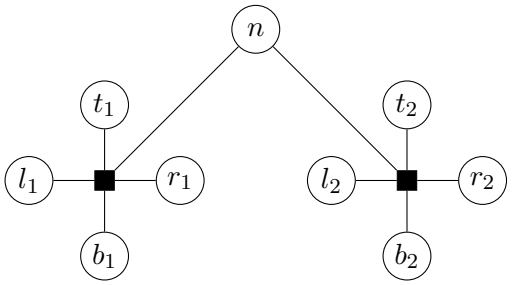


Figure 7.5 Possible CRF model for multiple bounding boxes. All variables are conditioned on the image but we have omitted x in the graph for clarity.

dummy values, indicating "no box".

7.6 Summary

We showed that the stochastic gradient approximations do not come as close to the true optimum within a given number of epochs as the LBFGS does in the same amount of iterations. However, for large values of λ, these methods converge faster during the first few epochs and stopping it early does not result in lower test performance. We showed that pseudolikelihood and piecewise training are well motivated for the object localization task. Furthermore, we gave directions for future work.

8

Conclusion

Previous methods for object localization lack a probabilistic interpretation. We have formulated the problem of object localization as a probabilistic model using the framework of conditional random fields where parameter learning can be performed by maximum conditional likelihood (MCL). Unfortunately, this is intractable in practice even for medium-sized images and datasets. We investigated four approximate learning methods, namely stochastic gradient descent (SGD), contrastive divergence (CD), pseudolikelihood (PL) and piecewise training (PW). We compared the learning methods against the exact MCL solution on the simple TU Darmstadt `cow` dataset. We then compared the three tractable methods CD, PL and PW against a structured support vector machine on the PASCAL VOC 2006 dataset.

The first comparison showed that the approximate methods resulted in parameters that generalized just as well as exact MCL. In the second experiment we trained CRF models for each of the 10 object categories in the dataset for each of the three tractable learning methods. This experiment showed that all methods achieved equal performance on the larger and more complex dataset. Furthermore, the methods obtained performance results comparable with those obtained by another struc-

tured output learning method, the structured support vector machine.

A

Structured support vector machine

In this appendix we describe the structured support vector machine and show how it can be used to learn the weights for the object localization problem. We show that ESS can be used to efficiently compute the prediction step during parameter learning as described by Blaschko and Lampert [6]. Finally, we compare parameter learning using the structured support vector machine with the maximum conditional likelihood learning of Section 3.4.

A.1 Structured SVM learning

In Chapter 3 we set up the problem of learning the parameters for the structured prediction model as maximizing the conditional likelihood of the weights given the data. Recall that we are interested in finding a function f that maximizes some quality function g:

$$f(x) = \arg\max_{y \in \mathcal{Y}} g(x, y) \tag{A.1}$$

Also recall that we defined $g(x, y) = \langle w, \varphi(x, y) \rangle$ for the object localization problem where $\langle w, \varphi(x, y) \rangle$ is a histogram of visual words and w is a weight vector that indicates how much the visual words belong

to the given object class.

An alternative view of this learning problem is minimizing some loss function over the entire dataset:

$$E_N(f) = \frac{1}{2}||w||^2 + C\sum_{n=1}^{N}\Delta(y^n, f(x^n)) \tag{A.2}$$

where N is the size of the dataset. This is known as the *regularized empirical risk* where $\Delta(y^n, y)$ is a loss function that measures how wrong the prediction y is from the ground truth and $\frac{1}{2}||w||^2$ ensures regularization of w. Since small changes in the weight vector w will not change the prediction $f(x)$, the loss $\Delta(y^n, f(x^n))$ is usually piecewise constant with respect to w and therefore not amenable to gradient descent methods. By adding a new set of variables $\xi = (\xi_1, \xi_2, \ldots, \xi_N) \in \mathbb{R}_+^N$ the empirical risk can be minimized using the *structured support vector machine* [6, 44, 58]:

$$\min_{w,\xi} \quad \frac{1}{2}||w||^2 + C\sum_{n=1}^{N}\xi_n \tag{A.3}$$

$$\text{s.t.} \quad \xi_n \geq 0, \quad \forall n \tag{A.4}$$

$$\langle w, \varphi(x^n, y^n)\rangle - \langle w, \varphi(x^n, y)\rangle \geq \Delta(y^n, y) - \xi_n, \quad \forall n, \ \forall y \in \mathcal{Y} \tag{A.5}$$

The structured SVM is a generalization of the multiclass SVM and we can interpret $\frac{1}{||w||}\left(\langle w, \varphi(x^n, y^n)\rangle - \langle w, \varphi(x^n, y)\rangle\right)$ as the margin. Because of the constraint (A.5) minimizing w thus corresponds to maximizing this margin.

In the standard multiclass SVM we usually have $\Delta(y^n, y) = \mathbb{1}_{[y^n \neq y]}$, the zero-one loss. However, this is not practical when dealing with structured output. For the object localization problem we want to give a higher loss to bounding boxes with less area overlap and zero loss to boxes with complete overlap. Blaschko and Lampert define the following loss function for the object localization problem:

$$\Delta(y^n, y) = 1 - \frac{\text{Area}(y^n \cap y)}{\text{Area}(y^n \cup y)} \tag{A.6}$$

where $\text{Area}(y^n \cap y)$ is the area of the intersection of the two boxes and $\text{Area}(y^n \cup y)$ is the area of the union of the two boxes. This is the

PASCAL measure for area overlap that we also used in Chapter 6 to measure the performance on the object localization task.

Since $|\mathcal{Y}|$ is very large, solving the above optimization is intractable because of the exponentially many constraints. However, we note that the $N|\mathcal{Y}|$ constraints in (A.5) are equivalent to the following N constraints:

$$\xi_n \geq \max_{y \in \mathcal{Y}} \left[\Delta(y^n, y) + \langle w, \varphi(x^n, y)\rangle - \langle w, \varphi(x^n, y^n)\rangle\right] \tag{A.7}$$

$$= -\langle w, \varphi(x^n, y^n)\rangle + \max_{y \in \mathcal{Y}} \left[\Delta(y^n, y) + \langle w, \varphi(x^n, y)\rangle\right], \quad \forall n \tag{A.8}$$

We note that this is very similar to computing the prediction (A.1) with the only difference being the added loss function. Therefore, we can utilize the ESS algorithm as discussed in Chapter 2 for efficiently computing this maximization [6].

It is beyond the scope of this thesis to discuss the dual formulation and max-margin learning algorithms but we refer the reader to the following texts on structured SVMs and max-margin learning [6, 44, 56, 57, 58].

A.2 Comparison with MCL

Instead of adding slack variables to the problem, another way to amend the non-differentiability of $\Delta(y^n, y)$ is to replace it by a convex upper bound $L(x^n, y^n, w)$, the optimization of which can be shown to yield the same solution w^*. This gives us the following optimization problem:

$$w^* = \arg\min_w \frac{1}{2}||w||^2 + C \sum_{n=1}^{N} L(x^n, y^n, w) \tag{A.9}$$

An upper bound for $\Delta(y^n, y)$ is

$$L_{\text{SSVM}}(x^n, y^n, w) = -\langle w, \varphi(x^n, y^n)\rangle + \max_{y \in \mathcal{Y}} \left[\Delta(y^n, y) + \langle w, \varphi(x^n, y)\rangle\right] \tag{A.10}$$

which is a generalization of the hinge loss used in multiclass SVMs.

Although we introduced maximum conditional likelihood learning from a probabilistic perspective in Chapter 3 we can also view it as

a loss minimization problem. In fact, the negative conditional log-likelihood can be viewed as a loss function:

$$L_{\mathrm{CRF}}(x^n, y^n, w) = -\log p(y^n|x^n, w) \tag{A.11}$$

$$= -\langle w, \varphi(x^n, y^n)\rangle + \log Z(x^n, w) \tag{A.12}$$

Inserting this into (A.9) and setting $C = 1$ we retrieve minimization of the negative conditional log-likelihood in equation (3.27) with $\lambda = \frac{1}{2}$.

Note the close similarity between equations (A.12) and (A.10). In the generalized hinge loss we perform a maximization (a prediction) over all $y \in \mathcal{Y}$ whereas in the log-loss we perform statistical inference to compute $\log Z(x^n, w)$ which is a sum over all $y \in \mathcal{Y}$. Blaschko and Lampert [6] showed that the ESS algorithm can be used to efficiently solve the prediction step exactly for the structured SVM solution to the object localization problem. In this thesis we show how to efficiently approximate $\log Z(x^n, w)$ for the CRF solution to the object localization problem.

Pletscher et al. [47] show a deeper connection between the max-margin and the maximum likelihood and present a model that jointly minimizes the log-loss and maximizes the margin.

B

Datasets

In this appendix we will describe the details of the two datasets we have used for our experiments, namely the TU Darmstadt `cow` dataset and the PASCAL VOC 2006 dataset.

B.1 TU Darmstadt cow dataset

The TU Darmstadt `cow` dataset is available from the PASCAL object recognition database[1]. It consists of 111 images with a single cow on each image with the location annotated. All cows have roughly the same scale and orientation (side view, facing left) and there are only three distinct backgrounds. Many of the cow images are quite similar to at least one other cow image in the database. Please see Figure B.1 for example images as presented at `http://pascallin.ecs.soton.ac.uk/challenges/VOC/databases.html#TUD`. For model selection we trained on a selection of 20 images from the training set and used the remaining 91 images for validation. The selected images used for training is shown in Table B.1. When a suitable regularization parameter was found, we retrained on the full training set.

[1] `http://pascallin.ecs.soton.ac.uk/challenges/VOC/`

Figure B.1 TU Darmstadt `cow` dataset example images.

Training images
cow-pic71-sml-lt
cow-pic121-sml-lt
cow-pic140-sml-lt
cow-pic142-sml-lt
cow-pic171-sml-lt
cow-pic211-sml-lt
cow-pic282-sml-lt
cow-pic301-sml-lt
cow-pic322-sml-lt
cow-pic402-sml-lt
cow-pic421-sml-lt
cow-pic432-sml-lt
cow-pic452-sml-lt
cow-pic510-sml-lt
cow-pic521-sml-lt
cow-pic541-sml-lt
cow-pic550-sml-lt
cow-pic571-sml-lt
cow-pic620-sml-lt
cow-pic661-sml-lt

Table B.1 Selection of images used for training during model selection.

B.2 PASCAL VOC 2006

The PASCAL Visual Object Classes Challenge (VOC) 2006 [17] dataset consists of 5304 images of natural scenes, containing 9507 annotated objects of 10 different categories. See Figures B.2 and B.3 for ex-

ample images as presented at `http://pascallin.ecs.soton.ac.uk/challenges/VOC/databases.html#VOC2006`.

The data has been split into 50% for training and validation, and 50% for testing. The distributions of images and objects by class are approximately equal across the training/validation and test sets. See Table B.2 for a summary of the statistics of the training, validation, and test data.

	Train		Val		Trainval		Test	
	Images	**Objects**	**Images**	**Objects**	**Images**	**Objects**	**Images**	**Objects**
Bicycle	127	161	143	162	270	323	268	326
Bus	93	118	81	117	174	235	180	233
Car	271	427	282	427	553	854	544	854
Cat	192	214	194	215	386	429	388	429
Cow	102	156	104	157	206	313	197	315
Dog	189	211	176	211	365	422	370	423
Horse	129	164	118	162	247	326	254	324
Motorbike	118	138	117	137	235	275	234	274
Person	319	577	347	579	666	1156	675	1153
Sheep	119	211	132	210	251	421	238	422
Total	1277	2377	1341	2377	2618	4754	2686	4753

Table B.2 Statistics of the PASCAL VOC 2006 dataset.

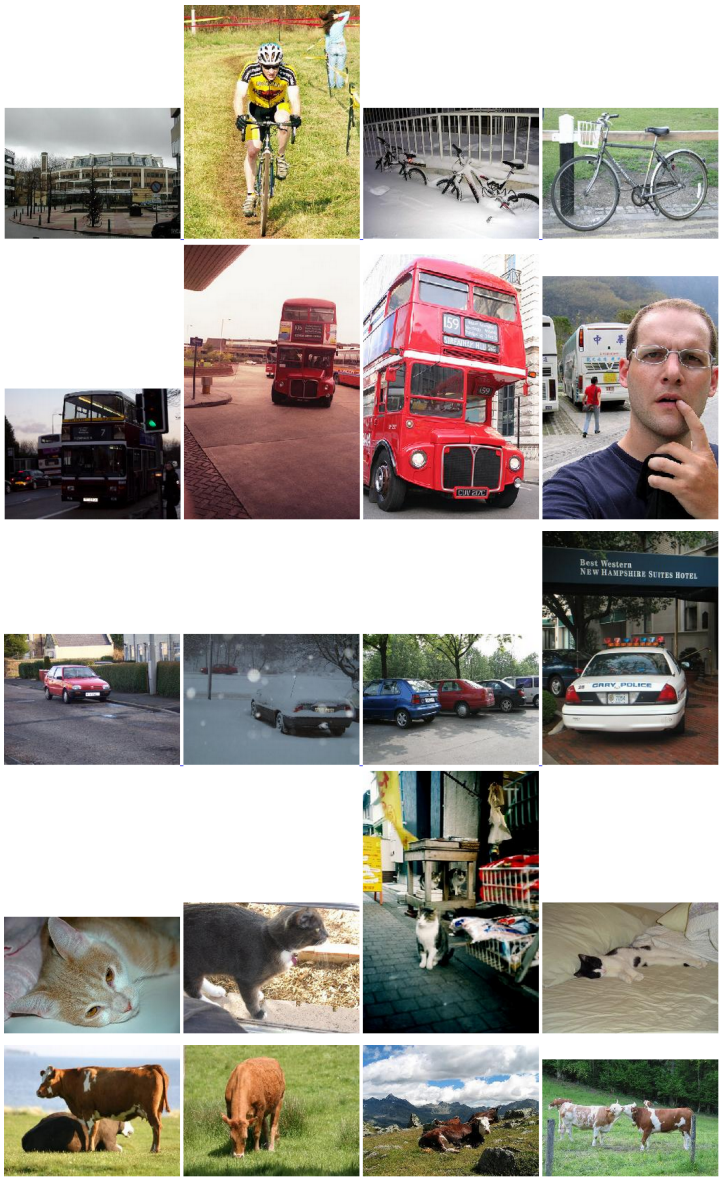


Figure B.2 PASCAL VOC 2006 example images: `bicycle`, `bus`, `car`, `cat`, `cow`.

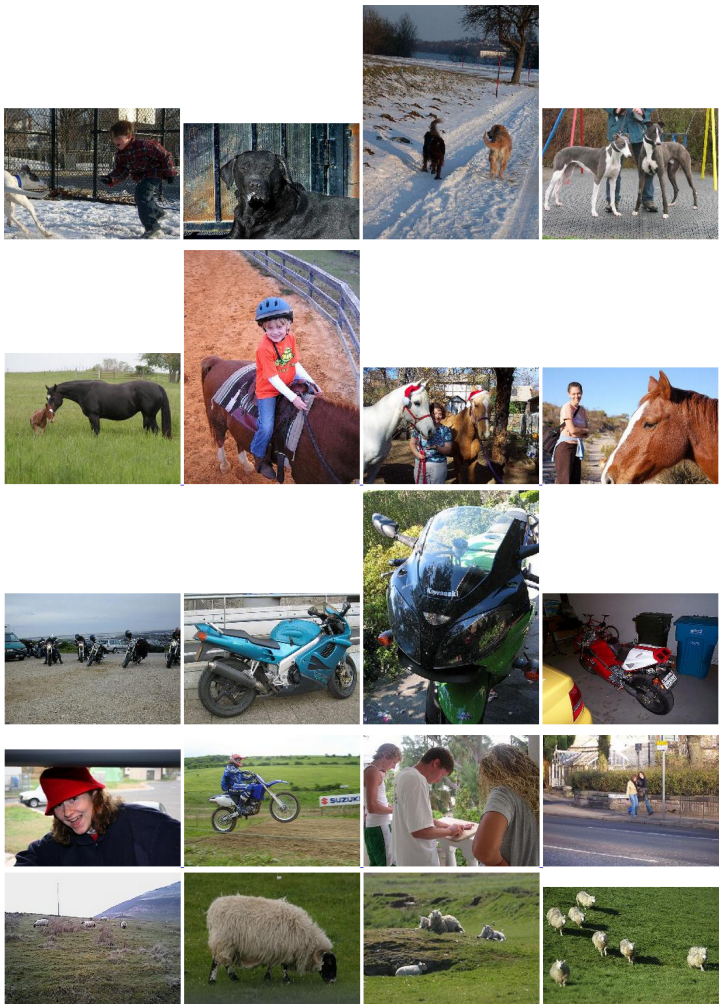


Figure B.3 PASCAL VOC 2006 example images: `dog`, `horse`, `motorbike`, `person`, `sheep`.

C

Choice of quantization step size

We explained in Section 3.6 that the height of an image H is a main contributor to high computational complexity as it enters into the computational complexity of maximum conditional likelihood learning as an $O(H^4)$ term. By quantizing the bounding box coordinates with some step size, we essentially reduce H by that step size and thereby obtain an increase in computational performance. With the introduction of the tolerance on the coordinate positions we, however, introduce a possible loss in localization performance.

In Chapter 6 we conducted experiments on the PASCAL VOC 2006 dataset (described in Appendix B) with a quantization of 8 on the coordinates. Since the images are medium-sized (resolution is about 500×500), we do not expect the objects within to be too small, and so we do not expect a quantization of 8 to introduce too big a loss. To convince ourselves that this postulate is reasonable, we conducted the experiments of contrastive divergence, pseudolikelihood and piecewise training on the PASCAL VOC 2006 dataset once again with a decreased quantization of 4 for the coordinates. The model parameter λ was found by performing model selection on the training and validation sets, and the chosen λ was then used for measuring the performance on the

CD on PASCAL VOC 2006			
object	λ	val	test
bicycle	100	0.532169	0.518738
bus	100	0.474421	0.507713
car	10	0.510348	0.452598
cat	100	0.567198	0.547331
cow	100	0.584496	0.52693
dog	10	0.499028	0.479088
horse	100	0.510186	0.488813
motorbike	10	0.518671	0.564208
person	10	0.284329	0.271184
sheep	100	0.456074	0.447437

Table C.1 Contrastive divergence overlap-precision: Validation and test performance for the best values of λ with a step size of 4.

test set. The results are listed for: contrastive divergence in Table C.1, pseudolikelihood in Table C.2, and piecewise training in Table C.3. We observe no significant difference neither between the methods all together at step size 4 nor when comparing to the results obtained with a step size of 8 in the Tables 6.2, 6.3, and 6.4. This justifies the choice of using a quantization step size of 8.

PL on PASCAL VOC 2006			
object	λ	val	test
bicycle	10000	0.538281	0.523584
bus	10000	0.474146	0.504196
car	10000	0.492735	0.448191
cat	10000	0.569289	0.546957
cow	10000	0.562867	0.517878
dog	10000	0.499218	0.491437
horse	10000	0.48974	0.482379
motorbike	10000	0.522343	0.563687
person	10000	0.26445	0.262026
sheep	10000	0.439028	0.439674

Table C.2 Pseudolikelihood overlap-precision: Validation and test performance for the best values of λ with a step size of 4.

PW on PASCAL VOC 2006			
object	λ	val	test
bicycle	10000	0.531432	0.529869
bus	10000	0.487147	0.510627
car	10000	0.527191	0.472532
cat	100000	0.574631	0.551673
cow	10000	0.574392	0.530932
dog	10000	0.504371	0.471695
horse	10000	0.476772	0.482334
motorbike	100000	0.500438	0.547296
person	100000	0.275632	0.290387
sheep	100000	0.485404	0.444804

Table C.3 Piecewise overlap-precision: Validation and test performance for the best values of λ with step size 4.

Bibliography

[1] H. Bay, T. Tuytelaars, and L. J. Van Gool. SURF: Speeded up robust features. In *European Conference on Computer Vision (ECCV)*, pages 404–417, 2006.

[2] Y. Bengio and O. Delalleau. Justifying and generalizing contrastive divergence. *Neural Computation*, 21(6):1601–1621, 2009.

[3] J. Besag. Statistical analysis of non-lattice data. *The Statistician*, 24(3):179–195, 1975.

[4] J. Besag. Efficiency of pseudolikelihood estimation for simple Gaussian fields. *Biometrika*, 64(3):616–618, 1977.

[5] C. M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, 2006.

[6] M. B. Blaschko and C. H. Lampert. Learning to localize objects with structured output regression. In *European Conference on Computer Vision (ECCV)*, pages 2–15, 2008.

[7] A. Bordes, L. Bottou, and P. Gallinari. SGD-QN: Careful quasi-Newton stochastic gradient descent. *Journal of Machine Learning Research (JMLR)*, 10:1737–1754, 2009.

[8] A. Bosch, A. Zisserman, and X. Munoz. Representing shape with a spatial pyramid kernel. In *ACM International Conference on Image and Video Retrieval (CIVR)*, pages 401–408, 2007.

[9] L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In S. Sra, S. Nowozin, and S. J. Wright, editors, *Optimization for Machine Learning*, pages 351–368. MIT Press, 2011.

[10] M. Carreira-Perpiñán and G. Hinton. On contrastive divergence learning. In *International Workshop on Artificial Intelligence and Statistics (AISTATS)*, pages 59–66, 2005.

[11] O. Chum and A. Zisserman. An exemplar model for learning object classes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2007.

[12] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *ECCV Workshop on Statistical Learning in Computer Vision*, pages 1–22, 2004.

[13] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 886–893, 2005.

[14] L. Devroye. *Non-Uniform Random Variate Generation*. Springer, New York, 1986.

[15] G. Dorkó and C. Schmid. Selection of scale-invariant parts for object class recognition. In *IEEE International Conference on Computer Vision (ICCV)*, pages 634–639, 2003.

[16] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision (IJCV)*, 88(2):303–338, 2010.

[17] M. Everingham, A. Zisserman, C. K. I. Williams, and L. Van Gool. The PASCAL Visual Object Classes Challenge 2006 (VOC2006) results. http://www.pascal-network.org/challenges/VOC/voc2006/results.pdf.

[18] L. Fei-Fei and P. Perona. A Bayesian hierarchical model for learning natural scene categories. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 524–531, 2005.

[19] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid. Groups of adjacent contour segments for object detection. *Pattern Analysis and Machine Intelligence (PAMI)*, 30(1):36–51, 2008.

[20] A. Fischer and C. Igel. Empirical analysis of the divergence of Gibbs sampling based learning algorithms for restricted Boltzmann machines. In *International Conference on Artificial Neural Networks (ICANN)*, pages 208–217, 2010.

[21] A. Fischer and C. Igel. Bounding the bias of contrastive divergence learning. *Neural Computation*, 23:664–673, 2011.

[22] A. Fischer and C. Igel. Training RBMs based on the signs of the CD approximation of the log-likelihood derivatives. In *European Symposium on Artificial Neural Networks (ESANN)*, 2011.

[23] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 6:721–741, 1984.

[24] B. Gidas. Consistency of maximum likelihood and pseudo-likelihood estimators for Gibbs distributions. *Stochastic Differential Systems, Stochastic Control Theory, and Applications*, 10:129–145, 1988.

[25] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer, second edition, 2009.

[26] W. K. Hastings. Monte carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[27] X. He, R. S. Zemel, and M. A. Carreira-Perpiñán. Multiscale conditional random fields for image labeling. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.

[28] G. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

[29] G. Hinton. A practical guide to training restricted Boltzmann machines. Technical Report UTML TR 2010-003, Department of Computer Science, University of Toronto, August 2010. Version 1.

[30] Q. Huang, M. Han, B. Wu, and S. Ioffe. A hierarchical conditional random field model for labeling and images of street scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1953–1960, 2011.

[31] A. Hyvärinen. Consistency of pseudolikelihood estimation of fully visible Boltzmann machines. *Neural Computation*, 18(10):2283–2292, 2006.

[32] D. Koller and N. Friedman. *Probabilistic Graphical Models - Principles and Techniques*. MIT Press, first edition, 2009.

[33] F. Korč and W. Förstner. Approximate parameter learning in conditional random fields: An empirical investigation. In *Symposium of the German Association for Pattern Recognition (DAGM)*, 2008.

[34] S. Kumar, J. August, and M. Hebert. Exploiting inference for approximate parameter learning in discriminative fields: An empirical study. In *Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR)*, pages 153–168, 2005.

[35] S. Kumar and M. Hebert. Discriminative Random Fields: A discriminative framework for contextual interaction in classification. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1150–1159, 2003.

[36] S. Kumar and M. Hebert. Discriminative random fields. *International Journal of Computer Vision (IJCV)*, 68(2):179–201, 2006.

[37] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *IEEE International Conference on Machine Learning (ICML)*, pages 282–289, 2001.

[38] C. H. Lampert. Kernel methods in computer vision. *Foundations and Trends in Computer Graphics and Vision*, 4(3):193–285, 2008.

[39] C. H. Lampert, M. B. Blaschko, and T. Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[40] D. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45:503–528, 1989.

[41] J. Nocedal. Updating quasi-Newton matrices with limited storage. *Mathematics of Computation*, 35:773–782, 1980.

[42] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, second edition, 2006.

[43] S. Nowozin, P. V. Gehler, and C. H. Lampert. On parameter learning in CRF-based approaches to object class image segmentation. In *European Conference on Computer Vision (ECCV)*, pages 98–111, 2010.

[44] S. Nowozin and C. H. Lampert. Structured learning and prediction in computer vision. *Foundations and Trends in Computer Graphics and Vision*, 6(3-4):185–365, 2010.

[45] S. Parise and M. Welling. Learning in Markov random fields: An empirical study. In *Joint Statistical Meeting (JSM)*, 2005.

[46] N. Plath, M. Toussaint, and S. Nakajima. Multi-class image segmentation using conditional random fields and global classification. In *IEEE International Conference on Machine Learning (ICML)*, pages 817–824, 2009.

[47] P. Pletscher, C. S. Ong, and J. M. Buhmann. Entropy and margin maximization for structured output learning. In *European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD)*, pages 83–98, 2010.

[48] A. Quattoni, M. Collins, and T. Darrell. Conditional random fields for object recognition. In *Advances in Neural Information Processing Systems (NIPS)*, volume 17, pages 1097–1104, 2005.

[49] A. Rakhlin, O. Shamir, and K. Sridharan. Making gradient descent optimal for strongly convex stochastic optimization. In *IEEE International Conference on Machine Learning (ICML)*, 2012. To appear.

[50] G. Roig, X. Boix, H. B. Shitrit, and P. Fua. Conditional random fields for multi-camera object detection. In *IEEE International Conference on Computer Vision (ICCV)*, pages 563–570, 2011.

[51] H. Rowley, S. Baluja, and T. Kanade. Human face detection in visual scenes. In *Advances in Neural Information Processing Systems (NIPS)*, volume 8, pages 875–881, 1996.

[52] J. Sivic, B. Russell, A. Efros, A. Zisserman, and W. Freeman. Discovering objects and their location in images. In *IEEE International Conference on Computer Vision (ICCV)*, volume 1, pages 370–377, 2005.

[53] C. Sutton and A. McCallum. Piecewise pseudolikelihood for efficient training of conditional random fields. In *IEEE International Conference on Machine Learning (ICML)*, pages 863–870, 2007.

[54] C. Sutton and A. McCallum. Piecewise training for structured prediction. *Machine Learning*, 77(2-3):165–194, 2009.

[55] C. Sutton and A. McCallum. An introduction to conditional random fields. In *Foundations and Trends in Machine Learning*, volume 4, pages 267–373, 2012.

[56] B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin. Learning structured prediction models: A large margin approach. In *IEEE International Conference on Machine Learning (ICML)*, pages 896–903, 2005.

[57] B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *Advances in Neural Information Processing Systems (NIPS)*, volume 16, 2003.

[58] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *IEEE International Conference on Machine Learning (ICML)*, pages 104–111, 2004.

[59] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *IEEE International Conference on Computer Vision (ICCV)*, pages 606–613, 2009.

[60] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 511–518, 2001.

[61] S. Vishwanathan, N. N. Schraudolph, M. W. Schmidt, and K. P. Murphy. Accelerated training of conditional random fields with stochastic gradient methods. In *IEEE International Conference on Machine Learning (ICML)*, pages 969–976, 2006.

[62] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.

[63] Y. Wang and S. Gong. Conditional random field for natural scene categorization. In *British Machine Vision Conference (BMVC)*, 2007.

[64] G. Winkler. *Image Analysis, Random Fields and Markov Chain Monte Carlo Methods: A Mathematical Introduction*. Springer, second edition, 2003.

[65] W. Xu. Towards optimal one pass large scale learning with averaged stochastic gradient descent. Technical report, 2010. arXiv:1107.2490v1.

[66] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13–13, 2006.