Image categorisation through Boosting using cost-minimising strategies for data labelling

Dissertation
Christian Savu-Krohn
July 21, 2009

1^{st} referee: Prof. Dr. Peter Auer, University of Leoben, Austria
2^{nd} referee: Prof. Dr. Günther Palm, University of Ulm, Germany

M. Comp. Sc. Christian Savu-Krohn
Image categorisation through Boosting using cost-minimising strategies for data labelling.
Dissertation,
University of Leoben, Austria.

to my parents

Contents

Acknowledgements

I would like to thank Peter Auer and Christos Dimitrakakis for their helpful discussions, suggestions and corrections. Furthermore I would like to thank my officemate Martin Antenreiter, my colleagues, Michael Kohn and Barbara Krenn for their support.

This work was supported in part by the Austrian Science Fund FWF (S9104-N13 SP4). The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7 / 2007-2013), PASCAL2, under grant agreement n° 216886. This publication only reflects the authors' views.

Abstract

Previous work from Opelt et al. [50, 49] and Fussenegger et al. [31] has shown that image categorisation using AdaBoost [29, 59] is a powerful method. They have used AdaBoost to select discriminative features to learn a classifier against a background class. As proposed in [3, 4] we present recent extensions to that framework by (a) incorporating geometric relations between features into the weak learner and (b) providing a weight optimisation method to combine pairwise classifiers for multiclass classification. We evaluate our framework on the Xerox data set [18] where we compare our results to the bag-of-keypoints approach. Moreover we report our results from the PASCAL Visual Object Classes Challenge 2006 [24].

The mass of images available through image databases, photo sharing websites, surveillance cameras a.s.o. is huge but obtaining the class information needed to learn a classifier is usually considered to be costly. One way to deal with the general problem of costly labels is *active learning*, where points to be labelled are selected with the aim of creating a classifier with better performance than that of a classifier trained on an equal number of randomly sampled points. Previous work [2, 14, 1, 13, 57, 43, 38, 6, 36, 19, 7] showed that active learning can improve the performance compared to standard passive learning. However the basic question of whether new examples should be queried at all is seldom addressed. This work deals with the labelling cost directly as recently proposed in [22]. The learning goal is defined as the minimisation of a cost which is a function of the expected model performance and the total cost of the labels used. This allows the development of general strategies and specific algorithms for (a) optimal stopping, where the expected cost dictates whether label acquisition should be terminated, and (b) empirical evaluation, where the cost is used as a performance metric for a given combination of learning, stopping and sampling methods. Though the main focus is optimal stopping, we also aim to provide the background for further developments and discussion within the field of active learning. Experimental results illustrate the proposed evaluation methodology and demonstrate the use of the introduced stopping method.

Nomenclature

a	weight of a classifier
β	exponential convergence parameter
c	class of convergence
C	random variable corresponding to the cost
D	slackness factor
VC	VC-dimension
ε	error given the training data
f	classifier
F	learning algorithm
g	error function
γ	query cost
h	convergence function
κ	quadratic convergence parameter
k_ϕ	number of clusters for feature type ϕ
λ	linear convergence parameter
m	number of examples
m_ϕ	total number of feature vectors extracted for feature type ϕ
m_t	number of observations of the error made per iteration
$m_\gamma\left(c, r_0, r_\infty\right)$	number of admissible convergence parameters at a cost-level of γ given c, r_0, r_∞
μ	margin
n	number of classes
ν	slackness parameter
OBSV	one-step bounded stopping algorithm with validation
ω	model
ϕ	feature type
q	realised error
$Q_F(\gamma)$	stopping algorithm
r	expected error
R	random variable corresponding to the classification error
r_0	initial error
r_∞	final error
ρ_0	bayes risk
ϱ	convergence parameter

`spam`	spambase database from `UCI`
t_γ	outcome for T_γ
$t_\gamma\left(c, \varrho, r_0, r_\infty\right)$	stopping time minimising the cost given $c, \varrho, r_0, r_\infty$ and γ
T_γ	random variable corresponding to the stopping time
θ	threshold
υ	reference feature
`UCI`	repository of machine learning databases from the University of California, Irvine
v_e	expected costs given a specific (training) dataset
`VOC06`	PASCAL Visual Object Classes Challenge 2006 data set
v	feature
w	weight of an example
`wdbc`	Wisconsin breast cancer data set from `UCI`
x	example
`Xerox`	Xerox data set
ξ	belief
y	class
z	observation of the error
ζ	slackness variable

Chapter 1

Image categorisation

1.1 Introduction

The field of Computer Vision recently has drawn the attention of many researchers. Today, their findings are fundamental to typical applications like driver assistance systems, robotics, surveillance and many others. According to [52] some of these applications represent an *object categorisation* problem:

> The research goals in object categorisation are to detect objects in images and to determine the objects categories. Categorisation aims for the recognition of generic classes of objects, and thus has also been termed 'generic object recognition'. This is in contrast to the recognition of specific, individual objects. ... Major problems are related to the concept of a 'visual category', where a successful recognition algorithm has to manage large intra-class variability versus sometimes marginal inter-class differences. ... We can define visual object categorisation as the process of assigning a specific object to a certain category.

For some applications it is sufficient to predict the occurrence of an object of a specific category while its localisation within the image or its shape are not of primary interest. This problem is termed *image categorisation*.

Image categorisation is usually performed upon real-world images including small or partially occluded objects that can be surrounded by background clutter. Therefore localisation and shape modelling become harder than in case of generic object recognition where target objects are usually more prominent with little variation in pose and the background is more homogeneous [27, 39, 28]. Although background information can contain valuable side information, for example asphalt points to a car rather than a horse, the question whether such classifiers can be learned using real-world images and if they really improve performance remains open.

Furthermore there exist many approaches to specific object categorisation problems like face detection [66], but the task becomes more difficult

if the goal is to develop a more general learning algorithm that can learn various target categories. One reason is that it is not always clear which types of features are advisable to learn a certain visual category. Thus even when most of the detected features are located on the target object, shape modelling may fail. These issues become even more important in case of a multiclass learning problem.

This work addresses those problems by extending a recent method from Opelt et al. [50, 49] and Fussenegger et al. [31]. They have shown that image categorisation using AdaBoost [29, 59] is a powerful method. In particular they have used AdaBoost to select discriminative features of various types to learn a classifier against a background class, which consists of similar scenes but without target objects or perceptually similar categories. In this chapter we introduce their framework in more detail (Sec. 1.3) and propose extensions to it along the way. More specifically, we will provide (a) a method to learn geometric relations between the locations of detected features (Sec. 1.3.3) and (b) a weight optimisation method to combine the predictions of the resulting pairwise classifiers for multiclass classification (Sec. 1.4). We conclude with an evaluation of our framework on the Xerox data set (`Xerox`) where we compare our results to the bag-of-keypoints approach (Sec. 1.5.2). Moreover we report our results from the PASCAL Visual Object Classes Challenge 2006 [24] (Sec. 1.5.3). Throughout this work we will use `VOC06` to refer to the corresponding data set whenever we conduct experiments not related to the challenge.

1.2 Related work

One recent approach to a general learning algorithm for various object categories is the "bag of keypoints" idea by Csurka et al. [18]. This method calculates a feature histogram for every image in the data set. Its main advantage is that standard learning algorithms like a SVM [35, 65], which need a fixed dimensional input vector, can be used to construct a classifier. On the other hand, using a fixed set of features for learning is critical as the features selected by the expert may lack discriminative information. Thus Farquhar et al. [25] proposed to make identification of suitable features a part of the learning process. However we argue that feature histograms cannot exploit geometric relationships between the features contained in an image although this might be discriminative information.

There exist various methods for incorporating such relationships between parts using statistical models. Early work in this direction was done by Burl et al. [11] for the recognition of planar object classes. There, important parts are selected by previously learned detectors, and afterwards a shape model is learned from the detected locations. This approach was later improved by using a soft-detection strategy in [12]. The two problems, detecting features

and building a shape model from the detection, are solved simultaneously. Later, unsupervised scale-invariant learning of parts and shape models has been done in [27], where an entropy-based feature detector from Kadir [37] has been used to select the important parts from an image.

Recently, graph-based models called "k-fans" were introduced [17]. The structure of the graph, and therefore the representational power of the shape model is controlled by the parameter k. There exist well defined algorithms to solve the learning and detection problems for models with k-fan graphs. In general, methods like [11, 12, 17, 27] force the user to predefine a fixed number of parts considered for learning. In consequence this quantity is usually determined using an independent validation set. In contrast, we show how to estimate the correct number of parts and how to learn the relative positions within the image at the same time using a boosting algorithm.

1.3 Categorisation of images through boosting

Fig. 1.1 shows an overview of the framework of Opelt et al. where our extensions to it are marked red. The training data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{m}$ consists of m labelled real-world images x_i where the label $y_i \in \{+1, -1\}$ indicates whether an object of a visual category is visible in the image or not. After preprocessing the images, different types of features are extracted from discontinuous or homogeneous regions using various existing methods (Sec. 1.3.1). Given those features a classifier is learned which predicts whether a visual category appears in an image or not. As it is not always clear beforehand which feature types are advisable for learning a certain category, Opelt et al. propose using all types simultaneously and to leave it up to the learning algorithm to determine the useful types.

In contrast to the original framework we suggest using LPBoost [9, 21] instead of AdaBoost since the former is more robust to noise (Sec. 1.3.2). Moreover, we propose incorporating geometric relations between features into the weak learner (Sec. 1.3.3). Our final extension to the original framework addresses the multiclass classification problem (Sec. 1.4) where we provide a weight optimisation method to combine pairwise classifiers by a SVM. For convenience the processing of an image by our framework will be illustrated by the running example shown in Fig. 1.2, an image from the visual category 'person' taken from the 'Graz-02' database set up by Opelt et al.[1].

1.3.1 Feature extraction

This section describes the methods used for feature extraction. As some of the feature extraction methods used here require grey-value images, we

[1] available at `http://www.emt.tugraz.at/~pinz/data/`

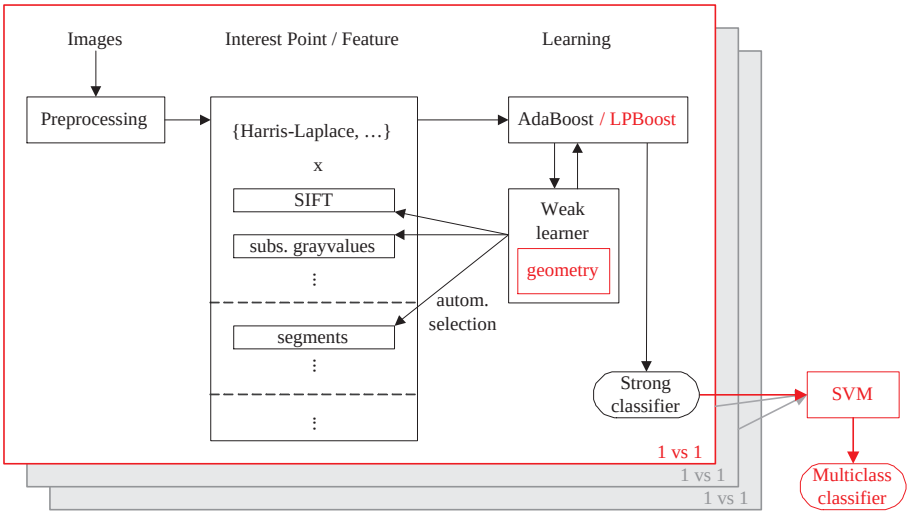


Figure 1.1: The image categorisation framework by Opelt et al. with our extensions marked in red. The weak learner of the boosting algorithm determines useful feature types automatically and searches for geometric relations between selected features. We suggest using LPBoost instead of AdaBoost to learn a pairwise classifier. The multiclass classification problem is addressed by combining such classifiers using a weight optimisation method.

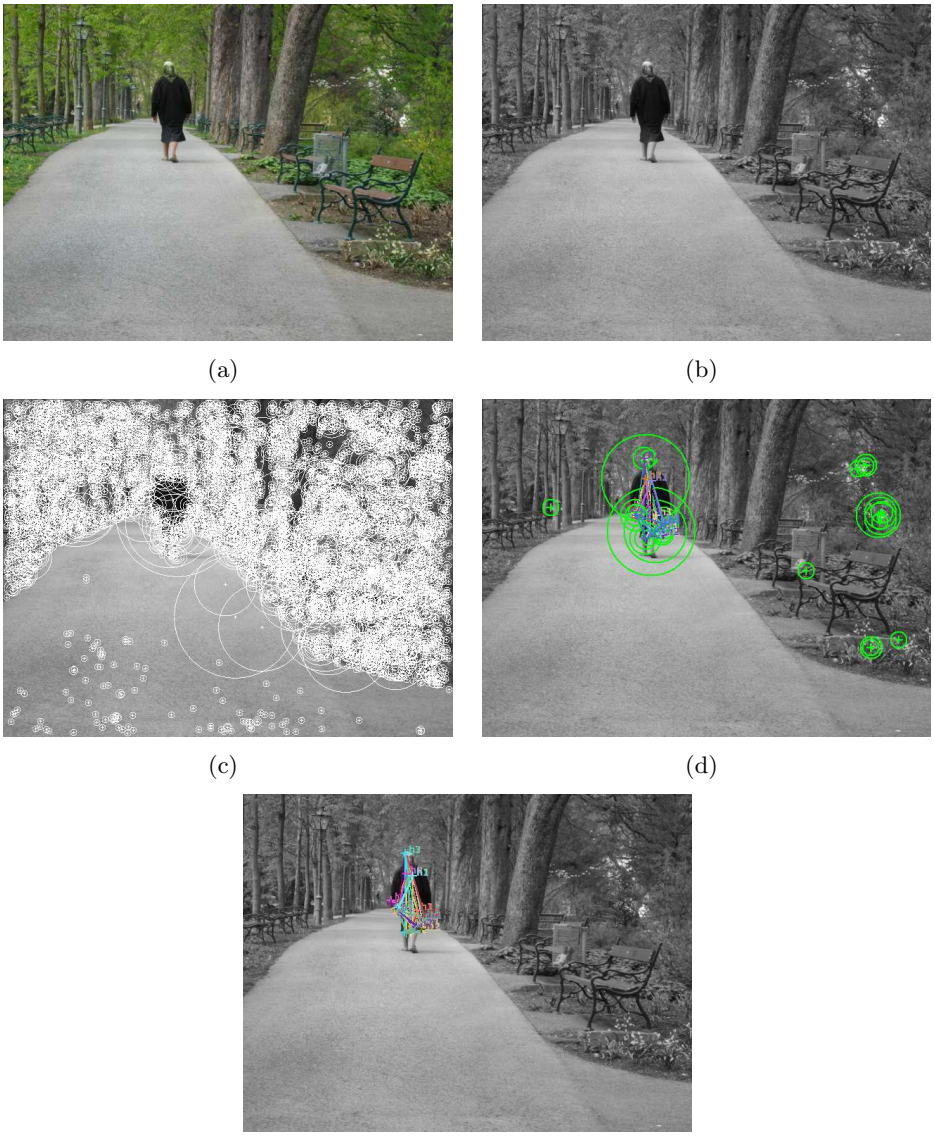
transform the images accordingly whenever appropriate (Fig. 1.2(a) - 1.2(b)). Then two groups of feature types are extracted from each image. The first describes various types of regions of discontinuity which have been located by various interest point detectors. The second group describes regions of homogeneity which are extracted by several segmentation methods. As a result of this process each image x_i will be represented by sets $\mathcal{V}_{i,\phi}$ of feature vectors $\mathbf{v}$, where $\phi = 1, 2, \ldots$ denotes the specific feature type as obtained from a specific combination of a localisation and a description method,

$$x_i \rightarrow \{\mathcal{V}_{i,1}, \mathcal{V}_{i,2}, \ldots\}\,. \tag{1.1}$$

Regions of discontinuity

The first step to extract regions of discontinuity is their localisation at salient events within an image. Based on a recent evaluation [61] the original framework proposes the use of the scale-invariant Harris-Laplace detector, the affine invariant interest point detectors, both proposed by Mikolajczyk and Schmid [44, 45], and the difference of Gaussian (DoG) detector as proposed by Lowe [40]. The interested reader may refer to [52] for a comprehensive overview on common detectors.

These methods utilise the concept of scale-space to detect interest points



(a) (b) (c) (d) (e)

Figure 1.2: An example of how an image is processed by our framework. We transform the original image (Fig. 1.2(a)) to grey-scale (Fig. 1.2(b)). Interest points (crosses) at various scales (circles) are detected and the corresponding features of various type ϕ are extracted (Fig. 1.2(c)). We obtain reference features by clustering each type. Fig. 1.2(d) shows weak classifiers (green marks) built by the boosting algorithm using a similarity threshold on the reference features in feature space, where it combines some of them geometrically. Useless weak classifiers will be discarded in subsequent iterations (Fig. 1.2(e)).

at their characteristic scale σ. The scale space of an image is obtained via a convolution of its intensity values $I(i,j)$ with a variable-scale Gaussian $G(I,\sigma)$ where σ denotes the standard deviation. Given such a space the goal is to extract visual features at their characteristic scale. For example applying the DoG function to the image given a certain value of σ emphasises the edges at this scale. The characteristic scale is then determined by searching for local maxima of the DoG over various scales [40]. Other approaches like the Harris-Laplace detector utilise more than one function to find an interest point. There the Harris function is used to localise interest points such as edges and corners given different values of σ, while the Laplacian response at those points is used to search for local maxima over the scales. As Mikolajczyk et al. have shown earlier [60] localisation using the Harris function is more reliable in the presence of image rotation and changes in illumination or perspective, while the Laplacian turns out to be most reliable in estimating the characteristic scale. Fig. 1.2(c) shows the interest points returned by this detector where the size of the surrounding circle indicates the characteristic scale. This approach was later extended toward affine-invariance of the detector [45]. More specifically, given an initial set of interest points obtained from a multi-scale Harris-detector, an iterative procedure is applied which modifies positions, scale and shape of the point neighbourhood, and converges toward a stable point that is invariant to affine transformations.

Given such detectors different types of features are extracted by applying different description methods to the image region around each interest point. According to a recent evaluation of different descriptors [46, 47], Opelt et. al suggest the use of sub-sampled grey-values (see [50]), basic intensity moments[2] and moment invariants [33] as well as scale invariant feature transforms (SIFTs) [40, 41]. We introduce these descriptors briefly.

One of the simplest methods to describe a region of interest is to subsample its grey-values using a window around each interest point. Although available information like the characteristic scale can be used to make it scale-invariant it is however not invariant to geometric distortions or changes in illumination.

Basic intensity moments are weighted averages of the pixel intensities of a region $I' \subset I$ in the image,

$$m_{p,q}^{k} = \sum_{i} \sum_{j} i^{p} j^{q} \left(I'(i,j) \right)^{k}$$

of order $p, q = 0, 1, 2, \ldots$ and degree $k = 0, 1, 2 \ldots$, and provide information about the centroid of the region and its orientation. These can be extended

[2] sometimes also referred to as raw intensity moment

to moment invariants like the central moment

$$\mu_{p,q} = \sum_i \sum_j \left(i - \frac{m_{1,0}}{m_{0,0}} \right)^p \left(j - \frac{m_{0,1}}{m_{0,0}} \right)^q I_{i,j}.$$

which is invariant to translation. Through combination of moments of different p, q one can attain further types of invariance. According to the work of Van Gool et al. [33], Opelt et. al suggest the use of four first order and five second order moment invariants which are affine and photometric invariant.

Finally Opelt et al. suggest the use of SIFT descriptors. Given an interest point together with its characteristic scale, a quadratic region around it is transformed w.r.t to this scale for normalisation. Next the region is normalized with respect to orientation. For this purpose a histogram for the 36 major directions of the local gradients at each pixel is calculated. The orientation of the region is then determined by the highest peak of that histogram. The region descriptor itself is calculated in a similar way. More specifically, the normalized region is divided into four non-overlapping quadratic subregions. For each subregion a histogram for the eight major directions $(0°, 45°, \ldots, 315°)$ of the gradients at each pixel is calculated. Thus the resulting descriptor is also partially invariant to illumination changes and affine or 3D projection.

It is important to note that subsampled grey-values, basic moments and moment invariants are not invariant to changes in illumination. Therefore Opelt et al. perform an intensity normalisation of the image regions for these three methods using homomorphic filtering (see for example [32], Chap. 4.9.6).

Regions of homogeneity

According to Opelt et al. regions of homogeneity are regions of limited difference of intensity values or regions with homogeneous texture. They suggest the use of two popular region-based segmentation algorithms: the Mean Shift algorithm [15] and their own method called Similarity-Measure-Segmentation [31]. Additionally we used the graph-based segmentation method by Felzenszwalb et al. [26] for some experiments.

The Mean Shift algorithm searches for modes of density over the joined space defined by pixel coordinates and intensity values. Instead of estimating the density itself, this method utilises a kernel density estimator [51][3] together with an efficient density gradient estimation technique. At each data point an iterative procedure is started which moves along the estimate of the current gradient until convergence at some stationary point, the representative of a cluster. Then clusters are merged if the distance of their

[3] also known as Parzen window

representatives is below the kernel bandwidth. Finally segments are obtained by labelling each cluster's support in the space of pixel coordinates uniquely.

Similarity-Measure-Segmentation extends the joint space of pixel coordinates and intensity values by textual information, high-pass, local binary patterns and wavelets, i.e. by information which accounts for the neighbourhood of a pixel as well. Given this space they calculate a similarity criterion based on a Gaussian kernel density function. Thereby the sensitivity of the similarity measure can be controlled by choosing a different bandwidth for the kernel along each dimension. Starting with a seeding pixel, this criterion is used to iteratively merge regions of pixels whose similarity is above a specified threshold until all pixels are labelled. Similarly to Mean Shift, all regions that are smaller than a specified value are eliminated by merging them with the spatially nearest region being larger than this value.

Felzenszwalb et al. [26] take a similar approach as they consider the joint feature space over spatial locations and intensity values as well. In contrast to Mean Shift and Similarity-Measure-Segmentation they utilise an undirected weighted graph to represent the similarity between points in this space. Then they apply a greedy strategy to search for a minimal cut given a parametrised merging criterion which trades off the *internal* dissimilarity of two segments versus the dissimilarity *across* both.

Given the segments obtained by Mean Shift and Similarity-Measure-Segmentation, Opelt et al. apply two kinds of description methods. The first describes the intensity values and their spatial distribution by their mean, variance, coefficient of variation, smoothness, skewness, kurtosis and the grey-value energy [34]. The second one contains invariant moments [42] which are invariant to scaling, rotation and translation, and are calculated from basic moments of inertia. It is important to note that Opelt et. al do not use colour information to describe segments although the Mean Shift algorithm and the Similarity-Measure-Segmentation are capable of such. Utilizing the fast segmentation method of Felzenszwalb et al. for segmentation, we therefore introduce a new description method where we characterise each segment by its colour information in LAB-space and its relative size compared to the image to which it belongs. We will refer to this type of features as colour segments.

1.3.2 Boosting algorithms

Boosting algorithms are a family of learning algorithms which combine a number of weak classifiers in an iterative process to improve overall classification performance. For this purpose they utilise a weak learning algorithm and call it on different weightings of the training data where the focus is set on examples that were hard to classify by the previous weak classifiers.

Algorithm 1 AdaBoost

Given a data set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^m$ with $y \in \{-1, +1\}$, a weak learning algorithm $WeakLearn$ and a maximum number of iterations J,

initialise $\mathbf{w} \leftarrow \left(\frac{1}{m} \ldots \frac{1}{m}\right)$
for $j = 1 : J$ **do**
  $f_j \leftarrow WeakLearn(D, \mathbf{w})$
  $\varepsilon_j \leftarrow \sum_{i=1}^{m} w_i \left[y_i \neq f_j\left(x_i\right)\right]$
  $a_j \leftarrow \frac{1}{2} \ln\left(\frac{1-\varepsilon_j}{\varepsilon_j}\right)$
  $w_i \leftarrow w_i \cdot \exp\left(-a_j y_i f_j\left(x_i\right)\right),\ i = 1, \ldots, m$
  $w_i \leftarrow w_i / ||\mathbf{w}||,\ i = 1, \ldots, m$
end for
return $f\left(x\right) = \operatorname{sgn}\left(\sum_{j=1}^{J} a_j f_j\left(x\right)\right)$

AdaBoost

A popular boosting algorithm is AdaBoost (Alg. 1). It reduces the training error on a given set of labelled examples exponentially fast as long the weak classifiers have a classification error $\varepsilon_j \leq 1/2 - \epsilon$ for some $\epsilon > 0$.

As shown by Alg. 1, initially the weights w_i of the examples are the same. At each round j AdaBoost calls the weak learning algorithm and measures the training error ε_j of the weak classifier f_j returned given the current example weights. In consequence f_j is assigned a fixed weight a_j within the ensemble, where $a_j \to 0$ as $\varepsilon_j \to 1/2$. Finally the weights for the misclassified examples are increased whereas those for correctly classified examples are decreased. Thus the misclassified examples will receive more attention by the weak learning algorithm in round $j + 1$. After the specified number of iterations J is reached AdaBoost returns the strong classifier

$$f(x) = sign\left(f^{\perp}(x)\right) = sign\left(\sum_{j=1}^{J} a_j f_j(x)\right) \in \{+1, -1\}, \qquad (1.2)$$

where the magnitude of the signed distance $f^{\perp}(x)$ of x to the decision boundary provides a measure of confidence with which f is making the prediction.

A common measure of performance is the *generalisation error*[4] of a classifier f given the data distribution $\mathscr{D}$ over $\mathcal{X} \times \mathcal{Y}$,

$$\mathbf{E}\left[R \mid f, \mathscr{D}\right] = \int_{\mathcal{X}} \sum_{y \in \mathcal{Y}} p\left(y \mid x, \mathscr{D}\right) \left[f(x) \neq y\right] p\left(x \mid \mathscr{D}\right)\, dx. \qquad (1.3)$$

Thus the *realised error* of a learning algorithm $F : \mathcal{D} \to \mathcal{F}$ given a specific

[4] sometimes also referred to as the *true error*

training set $\mathcal{D}$ drawn according to $\mathcal{D}$ is

$$q = \mathbf{E}\left[R \mid F, \mathcal{D}, \mathscr{D}\right] = \int_{\mathcal{F}} \mathbf{E}\left[R \mid f, \mathscr{D}\right] p(f \mid F, \mathcal{D})\, df. \tag{1.4}$$

where $\mathcal{F}$ is the space of classification functions[5] and the integral directly evaluates to $\mathbf{E}[R \mid f, \mathscr{D}]$ if F is deterministic.

Usually we only have access to a finite sample $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^m$ drawn i.i.d. according to $\mathscr{D}$ and therefore cannot observe the generalisation error directly. For this purpose we consult learning theory which analyses the factors that determine the generalisation error. Besides other approaches, one way to bound the generalisation error of a classifier is to adopt the notion of confidence, as given by Eq. 1.2, via the *(two-class hard) margin*,

$$\mu\left(x, y\right) = y f^{\perp}(x) = y \sum_{j=1}^{J} a_j f_j(x) \in [-1, 1]\,, \tag{1.5}$$

and the related *margin error* given the training data $\mathcal{D}$,

$$\mathbf{P}\left[\mu\left(x, y\right) \leq \theta\right] = \frac{1}{m} \sum_{i=1}^{m} \left[\mu\left(x_i, y_i\right) \leq \theta\right] \tag{1.6}$$

For weighted average functions f as such from AdaBoost it has been shown by Schapire et al. [58] that with probability at least $1 - \delta$,

$$\begin{aligned} \mathbf{E}\left[R \mid f, \mathscr{D}\right] \leq\ & \mathbf{P}\left[\mu\left(x, y\right) \leq \theta\right] + \\ & \mathcal{O}\left(\sqrt{\frac{1}{m}\left(\frac{VC\left(\mathcal{F}'\right) \cdot \log^2\left(m / VC\left(\mathcal{F}'\right)\right)}{\theta^2} + \log(1/\delta)\right)}\right), \end{aligned} \tag{1.7}$$

where $\theta > 0$ and $VC\left(\mathcal{F}'\right)$ is the VC-dimension of the space $\mathcal{F}'$ of weak classifiers with $m \geq VC\left(\mathcal{F}'\right) \geq 1$. This means that, under the assumptions made, the maximisation of the margins minimises the generalisation error. Furthermore Schapire et al. validated these theoretical findings using data with a small amount of noise, i.e. few outliers or mislabelled points. Their experiments showed that continued boosting after $\varepsilon \approx 0$ maximizes the margin of difficult examples, i.e. of those examples having *minimum margin*,

$$\mu = \min_{1 \leq i \leq m} \mu\left(x_i, y_i\right). \tag{1.8}$$

[5] also known as hypothesis space

LPBoost

While AdaBoost maximises the margin implicitly, *Linear Programming Boosting* (LPBoost, [21]) formulates margin maximisation explicitly. That is, the maximisation of the minimum margin becomes the objective of the linear optimisation problem (LP)

$$\begin{array}{lll} \max_{\mu,a} & \mu & \\ s.t. & y_i \sum_{j=1}^{|\mathcal{F}'|} a_j f_j(x_j) \geq \mu & i = 1, \ldots, m \\ & a_j \geq 0 & j = 1, \ldots, |\mathcal{F}'| \\ & \sum_{j=1}^{|\mathcal{F}'|} a_j = 1, & \end{array} \tag{1.9}$$

where the dual is given by

$$\begin{array}{lll} \min_{w,b} & b & \\ s.t. & \sum_{i=1}^{m} y_i w_i f_j(x_i) \leq b & j = 1, \ldots, |\mathcal{F}'| \\ & 0 \leq w_i & i = 1, \ldots, m \\ & \sum_{i=1}^{m} w_i = 1. & \end{array} \tag{1.10}$$

For data sets with a larger amount of noise however, increasing the smallest margin often degrades the generalisation error due to overfitting as shown by [53, 10, 54]. As pointed out by [54], maximisation of μ is indeed not obvious from the PAC-bound given in Eq. 1.7. Moreover, since the first term on the right hand side takes the *whole margin distribution* into account, allowing for a training error larger than zero leads to a larger value of θ, but therefore the second term can become smaller. Thus, a relaxation of the hard margin concept to allow for mistrusting parts of the data could be beneficial.

The concept of the *soft margin* was introduced by Cortes et al. [16] to improve the performance of the related SVMs given noisy data. More specifically, the soft margin of an example is defined by

$$\tilde{\mu}\left(x_i, y_i\right) = \mu\left(x_i, y_i\right) + D \cdot \zeta_i, \tag{1.11}$$

where ζ_i is a non-negative quantity expressing the mistrust in example x_i and D is an a priori chosen constant which trades off between margin maximisation and misclassification error. Accordingly, Demiriz et al. provide an alternative formulation for the LP given by Eq. 1.9 where the constraints are relaxed,

$$\begin{array}{lll} \max_{\mu,a,\zeta} & \mu - D \sum_{n=1}^{m} \zeta_i & \\ s.t. & y_i \sum_{j=1}^{|\mathcal{F}'|} a_j f_j(x_i) + \zeta_i \geq \mu & i = 1, \ldots, m \\ & \zeta_i \geq 0 & i = 1, \ldots, m \\ & a_j \geq 0 & j = 1, \ldots, |\mathcal{F}'| \\ & \sum_{j=1}^{|\mathcal{F}'|} a_j = 1, & \end{array} \tag{1.12}$$

and the dual LP is given by

$$\begin{array}{lll} \min_{b,w} & b & \\ s.t. & \sum_{i=1}^{m} y_i w_i f_j(x_i) \leq b & j = 1, \ldots, |\mathcal{F}'| \\ & 0 \leq w_i \leq D & i = 1, \ldots, m \\ & \sum_{i=1}^{m} w_i = 1. & \end{array} \tag{1.13}$$

It is instructive to note that in order to obtain any meaningful solutions the amount of slackness is controlled by a linear term within the objective of the primal. As can be seen by Eq. 1.13, the linear term and the relaxed constrains translate to an upper bound of D for the weights of the examples in the dual.

Demiriz et al. follow [55] and suggest controlling the generalisation capability according to $D = \frac{1}{\nu m}$ with $\nu \in (1/m, 1)$. More specifically, when $\nu \to 1$ the misclassification cost in the dual becomes the same for each example and only a few constraints will be active, i.e. equal b, within the solution. By the complementary slackness condition $a_j \left(\sum_{i=1}^{m} y_i w_i f_j(x_i) - b\right) = 0$ these correspond to those weak classifiers having $a_j > 0$ within the primal. Thus the resulting strong classifier combines only a few weak classifiers which usually results in a poor generalisation error. However as ν decreases the misclassification costs will increase for hard-to-classify points and will be zero for the others. Thus $\mathbf{w}$ becomes sparser and therefore the number of active constraints in the solution of the dual can increase. For $\nu \to 1/m$ however the number of $a_j > 0$ can become too large and the generalisation error of the strong classifier is poor due to overfitting to the training data. Thus ν is a critical parameter and therefore its optimal value is usually determined using a validation set.

For practical situations usually the dual problem (Eq. 1.13) is considered, since, in contrast to the primal, its solution does not require the generation of all $f' \in \mathcal{F}'$. More specifically, given a solution $(b, \mathbf{w})$ for a problem consisting of j weak classifiers, the task of the weak learning algorithm *WeakLearn*: $(\mathbf{w}, \mathcal{D}) \to \mathcal{F}'$ is to find a weak classifier f_{j+1} in the space of weak classifiers $\mathcal{F}'$ which maximises the accuracy under the current boosting weights,

$$f_{j+1} = \arg\max_{f' \in \mathcal{F}'} \sum_{i=1}^{m} y_i w_i f'(x_i) = \arg\max_{f' \in \mathcal{F}'} \frac{1 - \varepsilon'}{2}. \tag{1.14}$$

In fact, it is sufficient to either find a weak classifier for which

$$\sum_{i=1}^{m} y_i w_i f_{j+1}(x_i) > b$$

or guarantee that no such $f_{j+1} \in \mathcal{F}'$ exists. In the first case boosting is continued by adding the new weak classifier to the problem, i.e. by increasing j in Alg. 2. In the second case the current strong classifier f is optimal and

Algorithm 2 LPBoost

Given a data set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^m$ with $y \in \{-1, +1\}$, a weak learning algorithm $WeakLearn$ and a slackness parameter D,

initialise $\mathbf{w} \leftarrow \left(\frac{1}{m}, \ldots, \frac{1}{m}\right)$, $b \leftarrow 0$, $j \leftarrow 0$
loop
  $j \leftarrow j + 1$
  $f_j \leftarrow WeakLearn(\mathbf{w}, \mathcal{D})$
  if $\sum_{i=1}^{m} w_i y_i f_j(x_i) \leq b$ **then**
    $j \leftarrow j - 1$, **break**
  else

$$(\mathbf{w}, b) \leftarrow \begin{cases} \arg\min_{\mathbf{w},b} & b \\ s.t. & \sum_{i=1}^m w_i y_i f_k(x_i) \leq b \quad k = 1, \ldots, j \\ & 0 \leq w_i \leq D \qquad\quad i = 1, \ldots, m \\ & \sum_{i=1}^m w_i = 1 \end{cases}$$

    $\mathbf{a} \leftarrow$ Lagrangian multipliers from LP
  end if
end loop
return $f(x) = \operatorname{sgn}\left(\sum_{k=1}^{j} a_k f_k(x)\right)$

boosting can be stopped. In contrast to AdaBoost this provides a natural stopping criterion for the linear programming boosting (LPBoost) algorithm shown in Alg. 2.

1.3.3 Weak learner

Next we will describe the different types of weak learners used. The original framework by Opelt et al. suggests to build a weak classifier using a *reference feature vector* of some type together with a similarity threshold. Thus the prediction of a category is based on the appearance of such reference features within an image. We extend this idea and search for spatial geometric relations between the detected locations of reference features within the images.

Simple weak learner

For visual categorisation purposes we are interested in a set of reference features to represent the content of an image with respect to the target category[6]. For this the original framework suggests clustering the feature vectors for each type ϕ and to use the resulting sets of cluster centres $\Upsilon_\phi = \{\boldsymbol{\upsilon}\}$ for that purpose. Thus the weak learning algorithm has to determine

[6] such a set is sometimes also referred to as *visual vocabulary*

the most useful reference feature for classification given the current example weights. This is done as follows.

Let $\boldsymbol{\upsilon} \in \Upsilon_\phi$ be some distinctive reference feature vector with respect to the target category. Opelt et al. define its similarity to a feature vector $\mathbf{v}$ of same type ϕ by the euclidean distance

$$d(\mathbf{v}, \boldsymbol{\upsilon}) = \|\mathbf{v} - \boldsymbol{\upsilon}\|_2. \tag{1.15}$$

Thus if there is *any* $\mathbf{v} \in \mathcal{V}_{i,\phi}$ with a sufficiently low distance, the appearance of $\boldsymbol{\upsilon}$ in x_i, and therefore of the target category, is indicated. In order to select such distinctive reference feature vectors Opelt et al. consider the minimal euclidean distance between image x_i and some $\boldsymbol{\upsilon}$ of type ϕ,

$$d(x_i, \boldsymbol{\upsilon}) = \min_{\mathbf{v} \in \mathcal{V}_{i,\phi}} d(\mathbf{v}, \boldsymbol{\upsilon}) = \min_{\mathbf{v} \in \mathcal{V}_{i,\phi}} \|\mathbf{v} - \boldsymbol{\upsilon}\|_2 \tag{1.16}$$

and use a similarity threshold θ to define a *simple weak classifier*

$$f(x; \boldsymbol{\upsilon}, \theta) = \begin{cases} 1 & d(x, \boldsymbol{\upsilon}) \leq \theta, \\ -1 & else. \end{cases} \tag{1.17}$$

According to Eq. 1.14 we want θ to minimise the error given $\mathbf{w}$ and $\boldsymbol{\upsilon}$,

$$\theta = \arg\max_{\theta' \geq 0} \sum_{i=1}^{m} y_i w_i f(x_i; \boldsymbol{\upsilon}, \theta').$$

For this they sort the distances of $\boldsymbol{\upsilon}$ to the training images, i.e. let π be a permutation such that

$$d(x_{\pi(1)}, \boldsymbol{\upsilon}) \leq \cdots \leq d(x_{\pi(m)}, \boldsymbol{\upsilon}), \tag{1.18}$$

and determine the index s of the image at the optimal decision boundary given π and $\mathbf{w}$,

$$s = \arg\max_{s' \in \mathcal{S}'} \sum_{i=1}^{s'} y_{\pi(i)} w_{\pi(i)} \tag{1.19}$$

Finally they choose

$$\theta = \frac{d(x_{\pi(s)}, \boldsymbol{\upsilon}) + d(x_{\pi(s+1)}, \boldsymbol{\upsilon})}{2}.$$

Thus, given the multitude of reference features of various type ϕ the weak learner returns the weak classifier with lowest error (Eq. 1.14) and thereby automatically determines the useful ϕ. It is useful to note that one can calculate a sorted minimal distance matrix $\mathbf{D}^{m \times k_\phi}, k_\phi = |\Upsilon_\phi|$ according to Eq. 1.18 prior to boosting and thereby reduce the amount of memory to

represent the data to $\mathcal{O}(m \cdot k_\phi)$. Thus at every boosting round the computational effort is reduced to solving Eq. 1.19 which can be done efficiently in $\mathcal{O}(m \cdot k_\phi)$ time.

Fig. 1.2(d) visualises the detections by the simple weak classifiers that have been generated after multiple rounds of LPBoost. More specifically, the green crosses denote the image locations of interest points whose feature vectors triggered a simple weak classifier according to Eq. 1.17. The characteristic scales of these interest points are denoted by the surrounding green circles. The marks of different colour denote the detections by the geometric weak classifiers which we introduce next.

Geometric weak learner

The geometric weak learner we propose searches for spatial geometric relations between distinctive reference features. When searching for geometric relations the search space covers all detected locations and their relative positions. Thus a full search over all possible geometric directions is a computationally time consuming process. Therefore we use rather simple geometric relations. More precisely, we use the four geometric directions (up, down, left, right) by which we relate up to three reference features. We dub those *geometric primitives*. As learning some object category may require geometric relations that consist of more than three primitives, our search algorithm builds hierarchies of such relations as trees. These relations are referred to as 'relations A' throughout this work. We also consider more precise geometric primitives which distinguish between eight geometric directions. We refer to the corresponding relations as 'relations B'. Note that our geometric relations are invariant to translation and scale but not to rotation.

To speed up computation, our geometric weak learner uses a greedy search strategy to find geometric relations as shown by Alg. 3. More specifically, we combine the previous weak classifiers $f_1, \ldots, f_k, \ldots, f_{j-1}$ only with the best simple weak classifier $f(x; \boldsymbol{\upsilon}_\phi, \theta)$ (Alg. 3, step 3) given the current boosting weights. This is reasonable due to Eq. 1.14. Although there might exist a better weak classifier $\hat{f}^{AND}$, the search for it would require the combination of every reference feature of every type ϕ with each f_k. Nevertheless, we tested this search strategy on a subset of the data. More specifically, instead of f^{AND} as built in step 3, we generate an optimal weak classifier $\hat{f}_k^{AND}$ by choosing such an additional simple weak classifier $\hat{f}_k$ from all possible simple weak classifiers given $\Upsilon_1, \Upsilon_2, \ldots$, that the error of the combined hypothesis $\hat{f}_k^{AND} = f_k \; AND \; \hat{f}_k$ given the current example weights is minimised. However, as this approach yielded comparable results at higher computational cost, we omit it further on.

Fig. 1.3 illustrates the progress made by LPBoost using Alg. 3 in building a geometric relation. At each boosting round j, the weak learner ei-

Algorithm 3 Geometric weak learner.

Given a data set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{m}$ with $y \in \{-1, +1\}$ and associated example weights w_i, and previously generated weak classifiers $f_1, \ldots, f_{j-1}$,

1: Select weak classifier f' according to Eq. 1.14.
2: **for** $k = 1, \ldots, j-1$ **do**
3: Create a weak classifier using a logical AND: $f_k^{AND} \leftarrow f_k$ AND f'.
4: Calculate the set $\mathcal{X}' \subseteq \mathcal{X}$ of all images for which f_k^{AND} triggers.
5: The two weak sub-classifiers f_k, f' are applied on all $x \in \mathcal{X}'$ yielding two sets of detected locations per image. We select the most frequent geometric relation between these sets and build the *geometric weak classifier* f_k^{geom}.
6: **end for**
7: Compare performance of $f', f_1^{geom}, \ldots, f_{j-1}^{geom}$ according to Eq. 1.14 and output the best.

ther builds a simple or a geometric weak classifier. During the incremental construction of the geometric weak classifiers, various geometric weak sub-classifiers are generated. If such a weak sub-classifier is useful with respect to the final example weights, LPBoost incorporates it into the final decision function by assigning a positive weight a_j to it; otherwise a_j will be set to zero (see Fig. 1.2(e) for a geometric relation actually observed during the experiments). Hence, the final strong classifier can contain more than one geometric weak classifier per object. In consequence we do not have to flip input images such that the objects always face the same way (e.g. motorbikes, airplanes), but rather to ensure that there are sufficient examples for all the important orientations in the data set.

1.4 Multiclass image categorisation

Within our experiments for multiclass classification on `Xerox`, we observed large generalisation errors when using a common one-vs-all strategy or hierarchic classifiers. Considering the object categories of this database, it is likely that the extracted features are shared within different classes. Actually, Csurka et al. [18] do achieve good results learning feature histograms with a one-vs-all strategy. Nevertheless, feature histograms cannot exploit geometric relationships between the features contained in an image, although this might be discriminative information. Hence, we chose a pairwise strategy and combine our individual classifiers by a voting scheme.

Simple voting methods such as *majority voting using hard labels*,

$$y = \arg\max_{y':1,\ldots,n} \sum_{y''=1}^{n} f_{y',y''}(x),$$

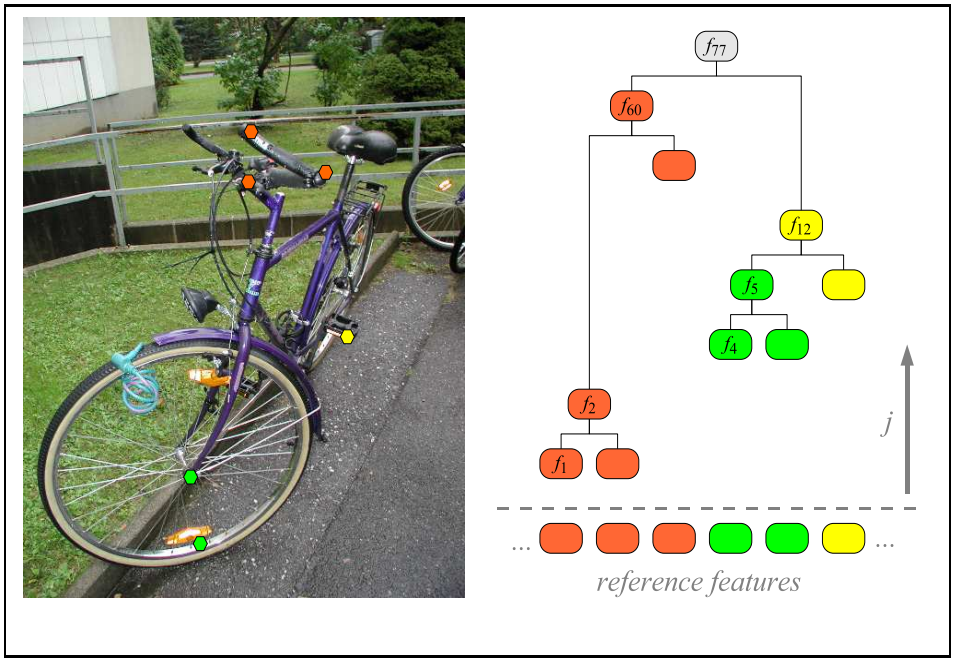


Figure 1.3: The development of a (fictional) hierarchy of geometric primitives by Alg. 3. At boosting iteration $j = 1$ the simple weak classifier f_1 consisting of an reference feature and a threshold is selected. It triggers for bar-like structures like the mounting point of the handle-bar. Next ($j = 2$) the geometric weak learner selects a similar simple weak classifier f' in *Step* 1 which triggers at the right hand of f_1. This geometric relation is most the frequent among all relations between the detections of f_1 and f', yielding f_2^{geom} (Step 5). Since f_2^{geom} performs performs better than f' it is selected as output (Step 7). With ongoing iterations it becomes more complex (f_{60}). Meanwhile other geometric weak classifiers like f_{12} were generated independently. At $j = 77$ however both get combined as indicated by f_{77}.

where the subscript y', y'' denotes a specific combination of target and counter class, not only ignore available information about the different degrees of *overall* confidence in some pairwise classifier, but also the classifier's *own* confidence in its prediction. Hence, a weighted voting scheme incorporating such information seems more reasonable.

According to Eq. 1.2 we propose measuring the confidence of a pairwise classifier $f_{y',y''}$ in its prediction by the signed distance $f^{\perp}_{y',y''}(x_i)$ of an example x_i to the decision boundary. Thus for an n-class problem we obtain $n \cdot (n-1)$ signed distances

$$\mathbf{f}^{\perp}(x_i) = \left(f^{\perp}_{1,2}(x_i), f^{\perp}_{2,1}(x_i), ..., f^{\perp}_{n-1,n}(x_i), f^{\perp}_{n,n-1}(x_i)\right)' \in [-1,1]^{n\cdot(n-1)\times 1}. \tag{1.20}$$

Using such we address the overall confidence in each classifier *given a certain class* $y \in \mathcal{Y}$. More specifically, for each $y_i \in \mathcal{Y}$ we try to find weights $\mathbf{w}_y \in \mathbb{R}^{1\times n\cdot(n-1)}$ together with a $b_y \in \mathbb{R}$ such that the overall vote corresponds to the true class y_i,

$$y_i = \arg\max_{y\in\mathcal{Y}} \mathbf{w}_y \cdot \mathbf{f}^{\perp}(x_i) + b_y. \tag{1.21}$$

For this purpose we formulate the following quadratic problem

$$\begin{array}{lll} \min & \| (\mathbf{w}_n, ..., \mathbf{w}_n) \|^2 + C \cdot \sum_i \zeta_i & \\ s.t. & \mathbf{w}_y \cdot \mathbf{f}^{\perp}(x_i) + b_y \geq 1 - \zeta_i, & y = y_i \\ & -\mathbf{w}_y \cdot \mathbf{f}^{\perp}(x_i) - b_y \geq 1 - \zeta_i, & \forall y \in \mathcal{Y} : y \neq y_i \\ & \zeta_i \geq 0 & i = 1, \ldots, m, \end{array} \tag{1.22}$$

which corresponds to a linear SVM. Similar to Eq. 1.12, the amount of slackness over all $\mathbf{f}^{\perp}(x_i)$ is controlled by the parameter C.

1.5 Experimental evaluation

Within this section we will evaluate our framework. First we describe the choice of the parameters for the feature extraction methods and the learning algorithms used (Sec. 1.5.1). Then we evaluate our framework on the `Xerox` data and compare our results to those reported in literature [18]. Thereby we also evaluate the improvements of our weighted voting approach over majority voting. Furthermore, we will report the results of our framework from the PASCAL Visual Object Classes Challenge 2006 (Eq. 1.5.3).

1.5.1 Parameter selection

For our experiments we use the Harris-Laplace detector to locate interest points and to extract regions of discontinuity of 16×16 pixels. To ensure at least one interest point per image we chose a minimum threshold with a value of 2900 for the response of the Laplacian and omit small regions with

a characteristic scale of $\sigma < 1.5$. Following Opelt et al. we extract four types of features from the regions of discontinuity: subsampled grey-values, basic moments, moment invariants and SIFTs.

For the first three feature types, we normalise illumination of the regions by homomorphic filtering, where we use a frequency threshold with a value of 2.1 below which values in the frequency domain are decreased by a factor of 0.6 and above which they are increased by a factor of 2. As it is not clear if discriminative information is lost by this normalisation we obtain three additional feature types using *unnormalized* image regions for these description methods. Furthermore, we use PCA to normalise all features types. Moreover we extract the SIFT descriptors using the binaries from D.G. Lowe[7]. Additionally we obtain another feature type by reducing these SIFT descriptors to their 40 largest components as obtained from PCA, which accounts for their sparseness.

According to the results reported by Opelt et al. Similarity-Based-Segmentation outperforms the Mean Shift approach. Therefore we use only the former method for segmentation with the parameter values suggested by Opelt et al., i.e. a value of 0.83 for the similarity threshold and of 50 for the minimum number of pixels in a region. Then we extract intensity values and their spatial distribution as described in Sec. 1.3.1 since Opelt et al. showed that this description method outperforms the method of invariant moments. We normalise these descriptors through whitening.

In a second preprocessing step we calculate the reference features. For this we cluster the different features by k-means using $k_\phi = |\Upsilon_\phi| = 2 \cdot \lfloor \sqrt{m_\phi} \rfloor$ centres with a random initialisation from the data, where

$$m_\phi = \left| \bigcup_{i=1}^{m} \mathcal{V}_{i,\phi} \right|$$

is the total number of feature vectors extracted.

1.5.2 Xerox database

We evaluate our image categorisation on the `Xerox` data. First we relate the performance of our multiclass framework to that reported for the bags-of-keypoints approach. Then we will investigate the geometric relations learned within the pairwise classifiers. Finally we will investigate the actual types of features learned by given different class combinations.

The Xerox database consists of 1774 real-world images from $n = 7$ different categories. The categories are faces (790), buildings (150), trees (150), cars (201), phones (216), bikes (125) and books (142). The numbers in parentheses indicate the number of images per category. Table 1.1 shows an

[7] available at `http://www.cs.ubc.ca/~lowe/keypoints/`

ϕ	**feature type**	**ill. norm.**	**PCA**	m_ϕ	k_ϕ
1	subs. grey-values	no	yes	854 376	1 848
2		yes	yes	854 376	1 848
3	basic moments	no	yes	852 755	1 846
4		yes	yes	854 376	1 848
5	moment invariants	no	yes	854 360	1 848
6		yes	yes	854 313	1 848
7	SIFTS	no	yes	809 063	1 798
8		no	PCA 40	809 063	1 798
9	segments	no	yes	690 070	1 661

Table 1.1: Feature types with normalisation steps for `Xerox`. For each feature type ϕ, m_ϕ denotes the total number of regions extracted from the data while k_ϕ denotes the number of reference features obtained by clustering the region descriptors with k-means.

overview of the extracted feature types where we note that the numbers of extracted feature vectors are considerable.

Due to time restrictions we determine the optimal slackness parameters D, C given the *whole* data set. More specifically we initially split the data into a training set and a test set, and utilise a simple iterative search using nested intervals over the parameter space which optimises the test error. We fix the optimal parameters. Then we perform a 10-fold stratified cross-validation to asses the sensitivity of the results to the data. For each run we split the data into a training set $\mathcal{D}$ and a test set $\mathcal{D}_E$. From $\mathcal{D}$ we select $n \cdot (n-1)$ training sets where each includes only the examples for a specific class combination. Given the fixed parameters we first train our pairwise classifiers and then utilise the weighted voting scheme[8] to obtain our multiclass classifier. Tab. 1.2 shows the average in realised accuracy, i.e. $1 - q$ (Eq. 1.4), as measured on $\mathcal{D}_E$. For reference Tab. 1.3 (top) shows the confusion matrix given the more complex 'relations B'.

As can be seen by Tab. 1.2 on average our weighted voting scheme outperforms standard majority voting by at least 16%. Furthermore our image categorisation framework outperforms the bag-of-keypoints approach of Csurka et al. by at least 5% on average, even without geometry. Although slight improvements can be observed for the more complex 'relations B', the difference is however not very significant as the confidence intervals overlap. Nevertheless the geometric relations learned seem reasonable as shown by Fig. 1.4, where we show exemplary images together with the detections of

[8]For the SVM we use SVM-light [35], available at `http://svmlight.joachims.org/`, where we also tried nonlinear kernels but omit their use further on since those kernels performed worse than the linear one.

F	ϕ	**voting**	**accuracy**
LPBoost + simple	$1 \dots 9$	majority	64.25 (3.21)
LPBoost + 'relations A'	$1 \dots 9$	majority	74.78 (2.92)
LPBoost + 'relations B'	$1 \dots 9$	majority	75.08 (2.51)
SVM + bag-of-keypoints [18]	7	$\max_y f_y^{\perp}(x)$	85.00 (n/a)
LPBoost + simple	$1 \dots 9$	weighted	90.60 (2.06)
LPBoost + 'relations A'	$1 \dots 9$	weighted	90.90 (2.16)
LPBoost + 'relations B'	$1 \dots 9$	weighted	**91.28** (2.28)

Table 1.2: Average accuracy as observed on `Xerox` from 10-fold stratified cross-validation given different types of learning algorithms and sets of feature types used.

the weak classifiers learned. All example images were taken from a single fold of the cross-validation procedure. For buildings vs. trees our method selects only one simple weak classifier using reference feature of type SIFT (PCA40) feature for classification. As can be seen by Fig. 1.4(a) - 1.4(c) it triggers particularly at window corners. Figures 1.4(d)-1.4(e) show false detections of that classifier since both images belong to the class of trees as those are in the foreground. Although the buildings are in the background, our classifier detects the window corners that are visible through and around the tree and is therefore still able to predict the building. On the other hand, Figure 1.4(f) gets misclassified as tree because there are no such corners visible. These examples show the difficulties in building an unambiguous database, and confirm the quality of our classifiers.

In that line of argument, one would expect that such a simple feature would be insufficient to distinguish buildings from books, since window corners are similar to corners of books. Indeed, the weak classifier of highest weight is a geometric relation between two features: one feature represents a window corner and the other triggers on green fields. The second best weak classifier uses three features, and votes in the case where there is a hedgerow in front of a building Fig. 1.4(g) - 1.4(i). This is reasonable considering that the class book contains only books on bookshelves or desktops, but no plants. Figures 1.4(j) and 1.4(k) belong to the class faces. The geometric weak classifier selected votes on triangle configurations of an ear, the hair line and the collar. Figures 1.4(l) and 1.4(m) show a weak classifier for the class of phones.

Furthermore, we analysed the overall selection of feature types by the total weight LPBoost assigns to the simple weak classifiers that utilize a certain feature type ϕ (Eq. 1.2). Given the optimal parameters, the initial split of the data and only simple weak classifiers (i.e. no geometry), it turned out that most pairwise classifiers select segments along with PCA40-

→	**faces**	**bldgs**	**trees**	**cars**	**phns**	**bikes**	**books**
faces	98.99	0.67	1.33	8.48	2.65	0	0.72
bldgs	0	70.67	8.00	0	0	2.84	8.93
trees	0	10.00	87.33	0	0	0.83	1.43
cars	0.51	0	0.67	84.10	9.42	0	0
phns	0.51	0	0	7.43	87.94	0	0
bikes	0	2.67	2.67	0	0	94.66	2.14
books	0	16.00	0	0	0	1.67	86.79

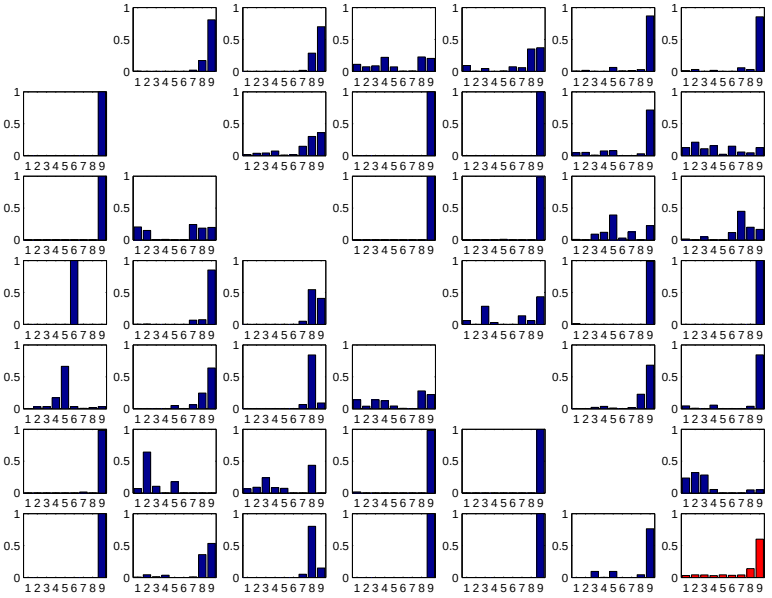


Table 1.3: **(top)** Confusion matrix as obtained from 10-fold cross validation using the more complex geometry 'relations B' and the weighted voting scheme. The true classes are denoted in the top row. **(bottom)** Total weight LPBoost assigns to each feature type ϕ given different class combinations of the initial split of the data for the non-geometric case. The lower right histogram shows the average in total weight over all class-combinations.

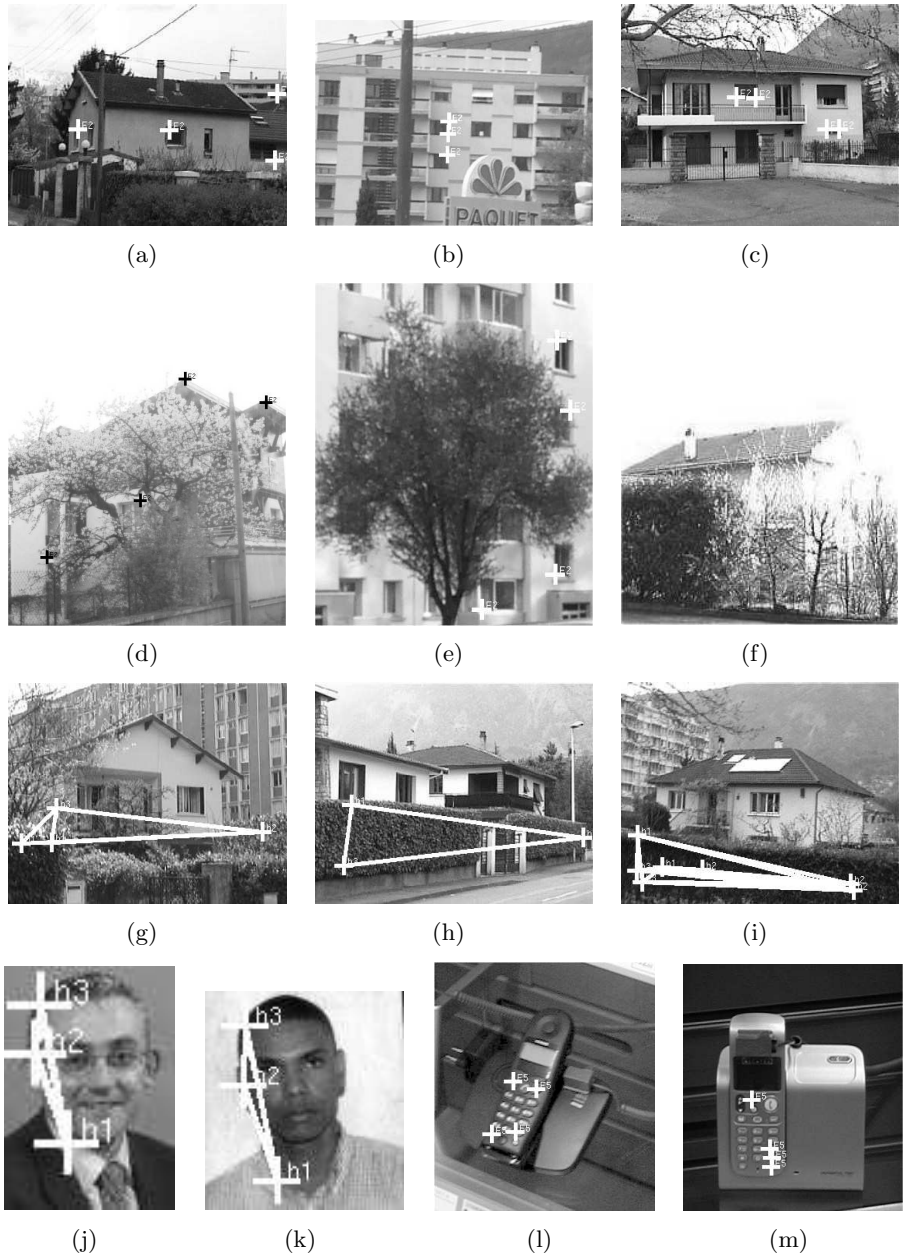


Figure 1.4: Example images with detections of selected weak classifiers. Fig. 1.4(a) - 1.4(f) are used for learning buildings vs. trees. Only the most important weak classifier and its matching feature locations are drawn. Fig. 1.4(a) - 1.4(c) show correct classifications, Fig. 1.4(d) - 1.4(f) show incorrect ones. Fig. 1.4(g) - 1.4(i) show correct classifications using a geometric classifier learned from buildings vs. books. Fig. 1.4(j) - 1.4(m) show geometric weak classifiers for faces and simple weak classifiers for phones.

SIFTs (Tab. 1.3). As shown in Table 1.3, for each pairwise classifier we observe a strong correlation between its average test error and the distribution of different feature types. That is, if two categories are hard to classify, the learning algorithm uses more feature types. This indicates the intrinsic flexibility of the framework when dealing with difficult class combinations. For example the results of Csurka et al. have shown that around 33% of all phones get misclassified by faces when learning is based solely on SIFT features. In contrast the misclassification rate of our method for this class combination is around 2.65% on average where our pairwise classifier utilises *various* feature types (Tab. 1.3). Although this specific result of Csurka et al. may have other reasons, we observed such improvements given other difficult class combinations as well.

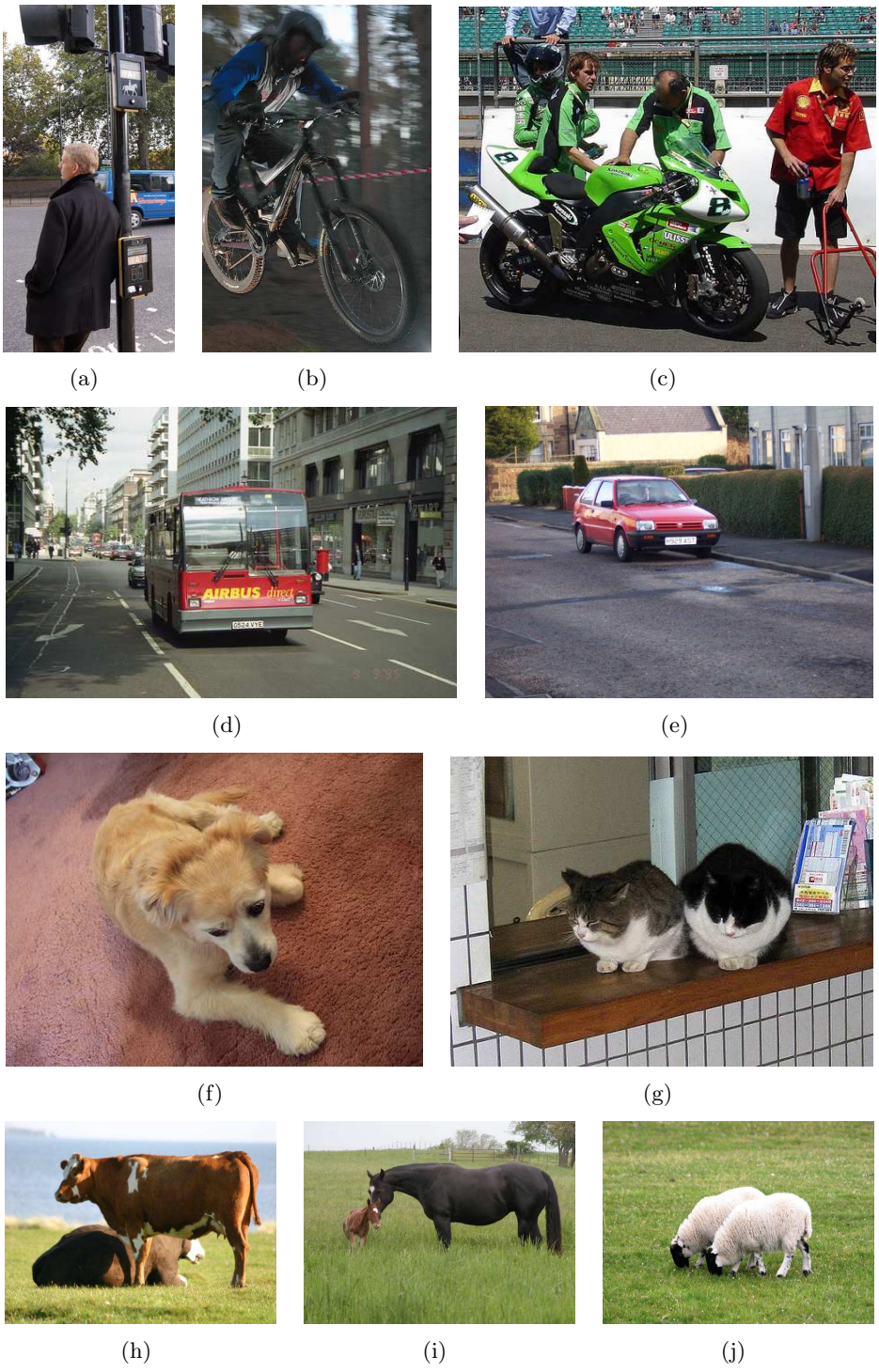
1.5.3 PASCAL VOC Challenge 2006

This section reports the results of our framework at the PASCAL VOC challenge 2006, where the goal was to categorise real-world images. The results for the participants were compared using the area under the ROC curve (AUC).

The data set for this challenge consists of 5304 images from $n = 10$ different classes covering bicycles (538), buses (354), cars (1097), cats (774), cows (403), dogs (735), horses (501), motorbikes (469), persons (1341) and sheep (489). The numbers in parentheses denote the number of images per category. As can be seen by the examples shown in Fig. 1.5 one image may contain objects of various classes.

The data set was split into 25% for training, 25% for validation and 50% for testing, where the labels for the latter were withheld until the submission deadline. For each target category the participants were free to submit their predictions using a separate learning method. More specifically, evaluation was based on the confidence value the classifier assigns to the occurrence of a target object of label y in a test image. The AUC for the target class was then calculated using different thresholds on such confidence values.

We split the training the data into $n \cdot (n-1)$ training sets where each set includes only the examples for a specific class combination. As an image can contain objects from more than one category, for pairwise learning we label an image as positive when an object of the target class is visible, irrespective of the appearance of the counter class. In contrast to the experiments on `Xerox`, we only used simple weak hypotheses since the computational effort to calculate geometric relations is huge given the number of extracted features and the number of classes. However, this time the slackness parameters D and C were optimised on the validation set, where C itself was optimised to maximise the AUC instead of the error. Thereby we use $\mathbf{w}_y \cdot \mathbf{f}^{\perp}(x_i) + b_y$ (see Eq. 1.21) as a measure of confidence in each $y \in \mathcal{Y}$ given x_i.



(a) (b) (c)

(d) (e)

(f) (g)

(h) (i) (j)

Figure 1.5: Example images from the PASCAL VOC Challenge 2006 for each class.

Within preliminary experiments we observed that our pairwise classifiers attain only moderate validation accuracy given class-combinations of cows, horses, sheep, cats and dogs. For this reason we decided to add the colour segments proposed in Sec. 1.3.1 to the set of feature types. More specifically, we used the binaries of Felzenszwalb et al.[9] and smoothed the images using a Gaussian filter as suggested choosing $\sigma = 0.3$. For the trade-off parameter of the merging criterion we used the default value of 500 and require a minimal segment size of 100. In addition to relative size w.r.t the image, we described these segments by their mean colour within the LAB-space relative to the standard D65 white point[10]. To preserve the perceptual uniformness of the LAB-space we calculate the similarity of two feature vectors $\mathbf{v}_\phi, \tilde{\mathbf{v}}_\phi$ of this type as follows. First we calculate their euclidean distance in LAB-space, and normalise the result using a logistic function,

$$d'(\mathbf{v}_\phi, \tilde{\mathbf{v}}_\phi) = 2 \cdot (0.5 + \exp\left(-\eta_1 \cdot \sqrt{(v_{\phi,L} - \tilde{v}_{\phi,L})^2 + (v_{\phi,a} - \tilde{v}_{\phi,a})^2 + (v_{\phi,b} - \tilde{v}_{\phi,b})^2}\right))^{-1},$$

with $\eta_1 = 0.02$. Then we penalise a large difference in segment size A by

$$d(\mathbf{v}_\phi, \tilde{\mathbf{v}}_\phi) = \begin{cases} d'(\mathbf{v}_\phi, \tilde{\mathbf{v}}_\phi), & \frac{\max(A,\tilde{A})}{\min(A,\tilde{A})} \leq \eta_2 \\ \frac{1}{\eta_2} \cdot \frac{\max(A,\tilde{A})}{\min(A,\tilde{A})} \cdot d'(\mathbf{v}_\phi, \tilde{\mathbf{v}}_\phi), & \frac{\max(A,\tilde{A})}{\min(A,\tilde{A})} > \eta_2 \end{cases}$$

with $\eta_2 = 4$. We use this measure for both, to cluster the features of this type (Sec. 1.5.1) and to calculate the minimal distance between an image x_i and a reference feature of this type (Eq. 1.16).

Tab. 1.4 shows an overview of the extracted feature vectors where we note that their numbers are considerable. Fig. 1.6 shows the total weight LPBoost assigns to simple weak hypotheses utilising reference features of specific types. As can be seen there for most class-combinations shown the total weight assigned to colour segments has either first or second rank amongst all feature types. Notably, the validation error of these pairwise classifiers improved compared to our preliminary experiments without colour segments. Thus we argue that the other feature types lack of discriminative information given these combinations of classes and that colour segments provide additional discriminative information. Moreover, given the average in total weight assigned over all class-combinations (as shown in the lower right) colour segments attain the second rank and therefore prove to be useful for other classes as well.

Tab. 1.5 shows the results of those participants of the challenge that achieve at least one top rank with respect to the AUC as measured on the

[9] available at `http://people.cs.uchicago.edu/~pff/segment/`

[10] ISO 10526:1999/CIE S005/E-1998, `http://www.cie.co.at/publ/abst/s005.html`

ϕ	**feature type**	**ill. norm.**	**PCA**	m_ϕ	k_ϕ
1	subs. grey-values	no	yes	4 017 292	4 008
2		yes	yes	4 017 292	4 008
3	basic moments	no	yes	4 017 292	4 008
4		yes	yes	4 017 292	4 008
5	moment invariants	no	yes	4 017 292	4 008
6		yes	yes	4 017 292	4 008
7	SIFTS	no	yes	7 132 276	5 341
8		no	PCA 40	7 132 276	5 341
9	segments	no	yes	2 838 449	3 369
10	colour segments	no	no	287 704	1 072

Table 1.4: Feature types with normalisation steps for VOC06. For each feature type ϕ, m_ϕ denotes the total number of regions extracted from the data while k_ϕ denotes the number of reference features obtained by clustering the region descriptors with k-means.

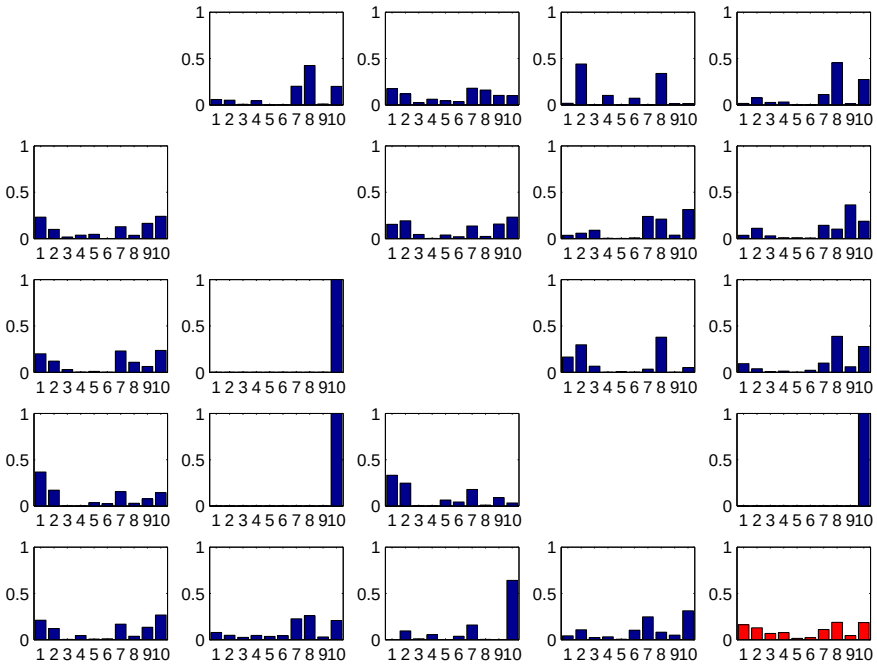


Figure 1.6: Total weight LPBoost assigns to each feature type ϕ given the combinations of the classes cats, cows, dogs, horses and sheep from the VOC06 database. The lower right histogram shows the average in total weight over the class-combinations given all $n = 10$ classes.

	INRIA		QMUL		XRCE	MUL
	Marsz.	Moosm.	HSLS	LSPCH		1vs1
bicycle	0.929	0.903	0.944	**0.948**	0.943	0.864
bus	**0.984**	0.933	**0.984**	0.981	0.978	0.945
car	0.971	0.957	**0.977**	0.975	0.967	0.928
cat	0.922	0.883	0.936	**0.937**	0.933	0.826
cow	0.938	0.895	0.936	0.938	**0.940**	0.789
dog	0.856	0.825	0.874	**0.876**	0.866	0.764
horse	0.908	0.824	0.922	**0.926**	0.925	0.733
motorbike	0.964	-	0.966	**0.969**	0.957	0.906
person	0.845	0.780	0.845	0.855	**0.863**	0.718
sheep	0.944	0.930	0.946	**0.956**	0.951	0.872

Table 1.5: Results of the PASCAL VOC challenge 2006. The columns show the AUC of all participants having at least one top rank (indicated by the bold letters) as measured on the test set given the target categories. The results of our framework are shown in the last column (MUL 1vs1).

test set given a specific target class. The results of our framework are shown in the last column. As can be seen there on average we obtain around 0.1 less AUC compared to the top rank of each class. Given a total of 20 submissions we achieve an average rank of 12.8 over all target classes. These results are reasonable as we (a) made little effort in tuning target-specific feature extractors, the number of clusters and so on, and (b) utilise the predictions of *only one* multiclass classifier to distinguish among *all* classes.

1.6 Discussion

This chapter discussed extensions to the image categorisation framework of Opelt et al.. Specifically we presented a method for learning geometric relations between features through boosting. Furthermore we provided a weighted voting scheme to combine the predictions of pairwise classifiers. To our knowledge this has not been done before.

During our experiments we found that learning without geometry already gives good performance, and that slight improvements are achieved by moving from simple weak classifiers to more complex geometric relations. An evaluation of the geometric weak classifiers obtained from the experiments on `Xerox` revealed that it is hard to find a relation with more than three features. Although the geometric relations learned are reasonable, simple weak classifiers using a single feature and pairwise geometric relations dominate the final solution. This might be due to the rather small

number of images for some classes since there seems not to be sufficient support within the images that would justify more geometric weak classifiers. For larger databases like `VOC06` however the computational effort to learn geometric relations becomes intractable due to the vast number of extracted features, many of them not located on the target object. Therefore one alternative would be to increase the amount of supervision by providing additional information about the location of target objects using bounding boxes for example. While the features extracted therein would be advisable to learn geometric weak classifiers, those located in the background can provide contextual information which could be learned using the simple weak learner. This however would shift the learning problem towards generic object recognition where object localisation matters.

Chapter 2

Cost-minimising strategies for data labelling

2.1 Introduction

It has become evident from the last chapter that much of classical machine learning deals with the case where we wish to learn a target concept in the form of a function $f : \mathcal{X} \to \mathcal{Y}$, when all we have is a finite set of examples $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^m$. However, in many practical settings, it turns out that only the observations x_i are available, while the availability of observations y_i is restricted in the sense that either (a) they are only observable for a subset of the examples or (b) further observations may only be acquired at a cost. This work deals with the second case, where we can actually obtain labels for any x_i, but doing so incurs a cost. Active learning algorithms (i.e. [14, 62]) deal indirectly with this by selecting examples which are expected to increase accuracy the most.

The potential of active learning is frequently showed by the following example. Assume we have data on the real line and we consider threshold functions of the form

$$f(x;\theta) = \begin{cases} 1, & x \leq \theta \\ 0, & else \end{cases}$$

Then classical learning theory tells us that if the underlying distribution $\mathscr{D}$ can be classified correctly by some $f \in \mathcal{F}$, the so-called realisable case, its enough to draw $m = \mathcal{O}\left(1/\epsilon \cdot \log(1/\epsilon)\right)$ random labelled examples from $\mathscr{D}$ and return any classifier consistent with them to achieve an error ϵ-close to zero. But if we draw m unlabelled examples from $\mathscr{D}$ we can infer an equally good classifier by a simple binary search that asks for only $\mathcal{O}\left(\log\left(1/\epsilon\right)\right)$ labels. Unfortunately this result does not hold in general. When we move to the unrealisable case, i.e. there is no $f \in \mathcal{F}$ that classifies $\mathscr{D}$ correctly, it has been shown by [36] that if the best $f \in \mathcal{F}$ has error $r > 0$ then the

label complexity becomes $\Omega\left(r^2/\epsilon^2\right)$ to obtain an accuracy ϵ-close to r. Thus active learning would essentially perform like passive learning.

However, the basic question of whether new examples should be queried at all is seldom addressed. This chapter deals with the labelling cost explicitly. We introduce a cost function that represents the trade-off between final performance (in terms of generalisation error) and querying costs (in terms of the number of labels queried). This is used in two ways. Firstly, as the basis for creating cost-dependent stopping rules. Secondly, as the basis of a comparison metric for learning algorithms and associated stopping algorithms.

To expound further, we decide when to stop by estimating the expected performance gain from querying additional examples and comparing it with the cost of acquiring more labels. One of the main contributions is the development of methods for achieving this in a Bayesian framework where we show experimentally that the stopping times we obtain are close to the optimal stopping times.

We also use the trade-off in order to address the lack of a principled method for comparing different active learning algorithms under conditions similar to real-world usage. For such a comparison a method for choosing stopping times independently of the test set is necessary. Combining stopping rules with active learning algorithms allows us to objectively compare active learning algorithms for a range of different labelling costs.

This chapter is organised as follows. Sec. 2.1.1 introduces the proposed cost function for when labels are costly, while Sec. 2.1.2 discusses related work. Sec. 2.2 derives a Bayesian stopping method that utilises the proposed cost function.

2.1.1 Combining classification error and labelling cost

There are many applications where raw data is plentiful, but labelling is time consuming or expensive. Classic examples are speech and image recognition, where it is easy to acquire hours of recordings, but for which transcription and labelling are laborious and costly. For this reason, we are interested in querying labels from a given data set such that we find the optimal balance between the cost of labelling and the classification error of the hypothesis inferred from the labelled examples. This arises naturally from the following cost function.

Let some algorithm F which queries labels for data from some unlabelled data set $\mathcal{D}$, incurring a cost $\gamma \in [0, \infty)$ for each query. If the algorithm stops after querying the labels of examples $x_1, x_2, \ldots, x_t$, with $t \in [1, |\mathcal{D}|]$ it will suffer a total cost of γt, plus a cost depending on the generalisation error. Let f_t be the classifier obtained after having observed t examples. Then we define the *total cost* for a specific classifier f_t as

$$\mathbf{E}[C_\gamma \mid f_t, \mathscr{D}] = \mathbf{E}[R_t | f_t, \mathscr{D}] + \gamma t. \tag{2.1}$$

We may use this cost as a way to compare learning and stopping algorithms, by calculating the expectation of C_γ conditioned on different algorithm combinations, rather than on a specific hypothesis. In addition, this cost function can serve as a formal framework for active learning.

Thus, these notions of optimality can in principle be used both for deriving stopping and sampling algorithms and for comparing them. Suitable metrics of expected real-world performance will be discussed in the next subsection. Stopping methods will be described in Sec. 2.2.

2.1.2 Related work

In the active learning literature, the notion of an objective function for trading off classification error and labelling cost has not yet been adopted. However, a number of both qualitative and quantitative metrics were proposed in order to compare active learning algorithms. Some of the latter are defined as summary statistics over some subset $\mathcal{T}$ of the possible stopping times. This is problematic as it could easily be the case that there exists $\mathcal{T}_1, \mathcal{T}_2$ with $\mathcal{T}_1 \subset \mathcal{T}_2$, such that when comparing algorithms over $\mathcal{T}_1$ we get a different result than when we are comparing them over a larger set $\mathcal{T}_2$. Thus, such measures are not easy to interpret since the choice of $\mathcal{T}$ remains essentially arbitrary. Two examples are (a) the *percentage reduction in error*, where the percentage reduction in error of one algorithm over another is averaged over the whole learning curve [57, 43] and (b) the average number of times one algorithm is significantly better than the other during an arbitrary initial number of queries, which was used in [38]. Another metric is the *data utilisation ratio* used in [38, 43, 1], which is the amount of data required to reach a specific error rate. It is instructive to note that the selection of the appropriate error rate is essentially arbitrary; in both cases the concept of the *target error rate* is utilised, which is the average test error when almost all the training set has been used.

Our setting is more straightforward, since we can use Eq. 2.1 as the basis for a performance measure. It is important to note that we are not strictly interested in comparing hypotheses f, rather algorithms F. In particular, we can calculate the expected cost given a learning algorithm F and an associated stopping algorithm $Q_F(\gamma)$, which is used to select the *stopping time* t_γ. From this follows that the *expected cost* of F when coupled with

$Q_F(\gamma)$ *given a data set* $\mathcal{D}$ sampled according to the data distribution $\mathscr{D}$ is

$$\begin{aligned} v_e(\gamma, Q_F(\gamma), F, \mathcal{D}, \mathscr{D}) &\equiv \mathbf{E}\left[C_\gamma \mid Q_F(\gamma), F, \mathcal{D}, \mathscr{D}\right] \\ &= \sum_t \left(\mathbf{E}[R_t \mid F, \mathcal{D}_t, \mathscr{D}] + \gamma t\right) \mathbf{P}(T_\gamma = t \mid Q_F(\gamma), F, \mathcal{D}) \\ &= \sum_t \left(\mathbf{E}[R_t \mid f_t, \mathscr{D}] + \gamma t\right) \mathbf{P}(T_\gamma = t \mid Q_F(\gamma), F, \mathcal{D}) \\ &= \sum_t \mathbf{E}[C_\gamma \mid f_t, \mathscr{D}] \, \mathbf{P}(T_\gamma = t \mid Q_F(\gamma), F, \mathcal{D}), \end{aligned} \tag{2.2}$$

where the sum directly evaluates to $v_e(\gamma, t_\gamma, F, \mathcal{D}, \mathscr{D})$ when $Q_F(\gamma)$ is deterministic. Furthermore we define the *expected error* of F at time t *given the data distribution* $\mathscr{D}$

$$r_t \equiv \mathbf{E}\left[R_t \mid F, \mathscr{D}\right] = \int_{S^t} \mathbf{E}\left[R_t \mid F, \mathcal{D}_t = u, \mathscr{D}\right] p(\mathcal{D}_t = u \mid F, \mathscr{D}) \, du = \mathbf{E}[q_t] \tag{2.3}$$

and the *expected cost given the data distribution* $\mathscr{D}$

$$\mathbf{E}\left[C_\gamma \mid Q_F(\gamma), F, \mathscr{D}\right] = \int_S \mathbf{E}[C_\gamma \mid Q_F(\gamma), F, \mathcal{D} = u, \mathscr{D}] p(\mathcal{D} = u \mid \mathscr{D}) \, du. \tag{2.4}$$

By keeping one of the algorithms fixed, we can vary the other in order to obtain objective estimates of their performance difference. In addition, we may want to calculate the expected performance of algorithms for a range of values of γ, rather than a single value, in a manner similar to what [8] proposed as an alternative to ROC curves. This will require a stopping method $Q_F(\gamma)$ which will ideally stop querying at a point that minimises $\mathbf{E}(C_\gamma)$.

The stopping problem is not usually mentioned in active learning literature and there are only a few cases where it is explicitly considered. One such case is [62], where it is suggested to stop querying when no example lies within the SVM margin. The method is used indirectly in [13], where if this event occurs the algorithm tests the current hypothesis[1], queries labels for a new set of unlabelled examples[2] and finally stops if the error measured thereby is below a given threshold; similarly, [6] introduced a bounds-based stopping criterion that relies on an allowed error rate. These are reasonable methods, but there exists no formal way of incorporating the cost function considered here within them. For our purpose we need to calculate the expected reduction in classification error when querying new examples and

[1] i.e. a classifier for a classification task

[2] Though this is not really an i.i.d. sample from the original distribution except when $|\mathcal{D}| - t$ is large.

compare it with the labelling cost. This fits nicely within the statistical framework of optimal stopping problems.

2.2 Stopping algorithms

An optimal stopping problem under uncertainty is generally formulated as follows [20]. At each point in time t, the experimenter needs to make a decision $a \in A$, for which there is a *loss function* $\mathcal{L}(a|\omega)$ defined for all $\omega \in \Omega$, where Ω is the set of all possible universes. The experimenter's uncertainty about which $\omega \in \Omega$ is true is expressed via the distribution $\mathbf{P}(\omega|\xi_t)$, where ξ_t represents his belief at time t. The *Bayes risk* of taking an action at time t can then be written as $\rho_0(\xi_t) = \min_a \sum_\omega \mathcal{L}(a,\omega)\,\mathbf{P}(\omega|\xi_t)$. Now, consider that instead of making an immediate decision, he has the opportunity to take *exactly* K more observations $\mathcal{D}_K$ from a sample space S^K, at a cost of γ per observation, thus allowing him to update his belief to $\mathbf{P}(\omega|\xi_{t+K}) \equiv \mathbf{P}(\omega|\mathcal{D}_K, \xi_t)$. What the experimenter must do in order to choose between immediately making a decision a and continuing sampling, is to compare the risk of making a decision now with the cost of making K observations plus the risk of making a decision after K time steps, when the extra data would enable a more informed choice. In other words, one should stop and make an immediate decision if the following holds for all K:

$$\rho_0(\xi_t) \leq K\gamma + \int_{S^K} p(\mathcal{D}_K = s|\xi_t) \min_a \left[\sum_\omega \mathcal{L}(a,\omega)\,\mathbf{P}(\omega|\mathcal{D}_K = s, \xi_t) \right] ds. \quad (2.5)$$

We can use the same formalism in our setting. In one respect, the problem is simpler, as the only decision to be made is when to stop and then we just use the currently obtained classifier. On the other hand for active learning we are interested in the more general problem where the experimenter may choose to stop at *any time* $t + k, 0 \leq k \leq K$ according to the observations made so far. The difficulty lies in estimating the expected error. Unfortunately, the metrics used in active learning methods for selecting new examples (see [38] for a review) do not generally include calculations of the expected performance gain due to querying additional examples.

There are two possibilities for estimating this performance gain. The first is an algorithm-independent method, described in detail in Sec. 2.2.1, which uses a set of learning curves, arising from theoretical convergence properties. We employ a Bayesian framework to infer the probability of each learning curve through observations of the error on the next randomly chosen example to be labelled. The second method, outlined in Chap. 4, relies upon a classifier with a probabilistic expression of its uncertainty about the class of unlabelled examples, but is much more computationally expensive.

Algorithm 4 A general bounded stopping algorithm using Bayesian inference.

Given a data set $\mathcal{D}$ and any learning algorithm F, an initial belief $\mathbf{P}(\omega \mid \xi_0)$ and a method for updating it, and additionally a known query cost γ, and a horizon K,

1: **for** $t = 1, 2, \ldots$ **do**
2: Use F to query a new example $i \in \mathcal{D}$ and obtain f_t.
3: Observe the empirical error estimate v_t for f_t.
4: Calculate $\mathbf{P}(\omega \mid \xi_t) \leftarrow \mathbf{P}(\omega \mid v_t, \xi_{t-1})$
5: **if** $\rho_K(\xi_t) > \rho_0(\xi_t)$ **then**
6: **break**
7: **end if**
8: **end for**
9: **return** t

2.2.1 Bayesian convergence estimation

The presented Bayesian formalism for optimal sequential decisions follows [20]. We require maintaining a belief ξ_t in the form of a probability distribution over the set of possible universes $\omega \in \Omega$. Furthermore, we require the existence of a well-defined cost for each ω. Then we can write the Bayes risk as in the left side of (2.5), but ignoring the minimisation over A as there is only one possible decision to be made after stopping,

$$\rho_0(\xi_t) = \mathbf{E}(R_t \mid \xi_t) = \sum_{\omega \in \Omega} \mathbf{E}(R_t \mid \omega)\, \mathbf{P}(\omega \mid \xi_t), \tag{2.6}$$

which can be extended to continuous measures without difficulty. We will write the expected risk according to our belief at time t for the optimal procedure taking at most k more samples as

$$\rho_{k+1}(\xi_t) = \min \left\{ \rho_0(\xi_t), \mathbf{E}[\rho_k(\xi_{t+1}) \mid \xi_t] + \gamma \right\}. \tag{2.7}$$

This implies that at any point in time t, we should ignore the cost for the t samples we have paid for and are only interested in whether we should take additional samples. The general form of the stopping algorithm is defined in Alg. 4. Note that the horizon K is a necessary restriction for computability. A larger value of K leads to potentially better decisions, as when $K \to \infty$, the bounded horizon optimal decision approaches that of the optimal decision in the unbounded horizon setting, as shown for example in Chapter 12 of [20]. Even with finite $K > 1$, however, the computational complexity is considerable, since we will have to additionally keep track of how our future beliefs $\mathbf{P}(\omega \mid \xi_{t+k})$ will evolve for all $k \leq K$.

Algorithm 5 OBSV, a specific instantiation of the bounded stopping algorithm.

Given a data set $\mathcal{D}$ with examples from the classes $\mathcal{Y}$ and any learning algorithm F, an initial belief $\mathbf{P}(\omega \mid \xi_0)$ and a method for updating it, and additionally a known query cost γ for discovering the class label $y_i \in \mathcal{Y}$ of example $x_i \in \mathcal{D}$,

1: Split $\mathcal{D}$ into $\mathcal{D}_A$ and $\mathcal{D}_R$,
2: $r_0 \leftarrow 1 - 1/|\mathcal{Y}|$.
3: $\mathcal{D}_T \leftarrow \emptyset$.
4: Initialise the classifier f.
5: **for** $t = 1, 2, \ldots$ **do**
6: Sample $(x_i, y_i) \in \mathcal{D}_R$ without replacement, observe $z_t \leftarrow [f(x_i) \neq y_i]$.
7: Calculate $\mathbf{P}(\omega \mid \xi_t) \leftarrow \mathbf{P}(\omega \mid z_t, \xi_{t-1})$.
8: If $\mathcal{D}_A \neq \emptyset$, set $k \leftarrow 2$, otherwise $k \leftarrow 1$.
9: **if** $\mathbf{E}[R_{t+k} \mid \xi_t] + k\gamma > \mathbf{E}[R_t \mid \xi_t]$ **then**
10: **break**
11: **end if**
12: If $\mathcal{D}_A \neq \emptyset$, use F to query a new example $(x_j, y_j) \in \mathcal{D}_A$ without replacement, $\mathcal{D}_T \leftarrow \mathcal{D}_T \cup (x_j, y_j)$.
13: $\mathcal{D}_T \leftarrow \mathcal{D}_T \cup (x_i, y_i)$, $f \leftarrow F(\mathcal{D}_T)$.
14: **end for**
15: **return** t

2.2.2 The OBSV algorithm

In this work we consider a specific one-step bounded stopping algorithm that uses independent validation examples for observing the empirical error estimate v_t, which we dub OBSV and is shown in detail in Alg. 5. The algorithm considers hypotheses $\omega \in \Omega$ of the learning curve which model how the expected error r_t of the learning algorithm changes with time. Specifically we assume that the *initial error* is r_0 and that the algorithm always converges to some unknown *final error* $r_\infty \equiv \lim_{t \to \infty} r_t$ at some unknown convergence rate. To model the convergence we will use different classes of theoretical convergence curves, that we index by c, along with a class-specific convergence rate parameter ϱ. Thus we consider models

$$\omega = (c, \varrho, r_0, r_\infty) \in \Omega. \tag{2.8}$$

where v_t will allow us to update our beliefs over Ω. The remainder of this section discusses the algorithm in more detail.

Steps 1 - 6, 12 - 13. Initialisation and observations

We begin by splitting the training set $\mathcal{D}$ in two parts: $\mathcal{D}_A$, which will be sampled without replacement by the *active learning algorithm* (if there is

one) and $\mathcal{D}_R$, which will be *uniformly* sampled without replacement. This condition is necessary in order to obtain i.i.d. samples for the inference procedure outlined in the next section. However, if we only sample randomly, and we are not using an active learning algorithm then we do not need to split the data and we can set $\mathcal{D}_A = \emptyset$.

At each time step t, we will use a sample from $\mathcal{D}_R$ to update $p(\omega)$. If we then expect to reduce our future error sufficiently, we will query an example from $\mathcal{D}_A$ using F and subsequently update the classifier f with both examples. Thus, not only are the observations used for inference independent and identically distributed, but we are also able to use them to update the classifier f.

Step 7. Updating the belief

We model the learning algorithm as a process which asymptotically converges from r_0 to r_∞. Each model ω will be a *convergence estimate*, a model of how the error converges from the initial to the final error rate. More precisely, each pair (c, ϱ) parametrises a function $h(t; c, \varrho) : \mathbb{N} \times \mathbb{R}^+ \to [0, 1]$ that models how close we are to convergence at time t. The predicted error at time t given ω will be

$$g(t; \omega) = g(t; c, \varrho, r_0, r_\infty) = r_\infty + (r_0 - r_\infty) \cdot h(t; c, \varrho). \tag{2.9}$$

We may now use these predictions together with some observations to update $p(\omega|\xi_{t-1})$. More specifically, if $\mathbf{P}[r_t = g(t; \omega) \mid \omega] = 1$ and we take m_t independent observations

$$\mathbf{z}_t = (z_t(1), z_t(2), \ldots, z_t(i), \ldots, z_t(m_t)) \in \{0, 1\}^{m_t},$$

where $z_t(i) = [f_t(x_i) \neq y_i]$, of the error with mean v_t, the likelihood of $\mathbf{z}_t$ will be given by the Bernoulli density

$$p(\mathbf{z}_t \mid \omega) = \left(g(t; \omega)^{v_t} [1 - g(t; \omega)]^{1-v_t}\right)^{m_t}. \tag{2.10}$$

Then it is simple to obtain a posterior density for ω,

$$p(\omega \mid \mathbf{z}_t, \xi_{t-1}) = \frac{p(\mathbf{z}_t \mid \omega) p(\omega \mid \xi_{t-1})}{p(\mathbf{z}_t)}. \tag{2.11}$$

Starting with a prior distribution $p(\omega \mid \xi_0)$, we may sequentially update our belief using Eq. 2.11 as follows:

$$p(\omega \mid \xi_t) \equiv p(\omega \mid \mathbf{z}_t, \xi_{t-1}). \tag{2.12}$$

The realised learning curve for a particular data set may differ substantially from the expected learning curve: the average learning curve will be smooth, while any specific instantiation of it will not be. More formally,

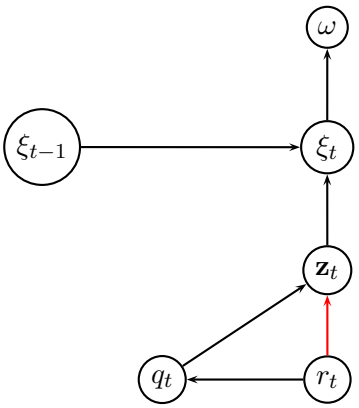


Figure 2.1: Dependency graph for updating the belief. At time t the current classifier has expected error q_t and we observe $\mathbf{z}_t$ of q_t. Together with ξ_{t-1} it then updates its belief to ξ_t about the models $\omega \in \Omega$ for the expected error r_t. According to Eq. 2.13 we can estimate a distribution for r_t from $\mathbf{z}_t$ without having to also estimate a distribution for q_t.

according to Eq. 1.4 the realised error of F given a specific training set is $q_t \equiv \mathbf{E}\left[R_t \mid F, \mathcal{D}_t, \mathscr{D}\right]$, while the *expected error* of F *given the data distribution* $\mathscr{D}$ is $r_t \equiv \mathbf{E}\left[R_t \mid F, \mathscr{D}\right]$. The smooth learning curves that we model would then correspond to models for r_t.

Fortunately, in our case we can estimate a distribution over r_t without having to also estimate a distribution for q_t, as this is integrated out for observations $z \in \{0, 1\}$

$$
\begin{aligned}
p(z \mid q_t) &= q_t^z \, (1 - q_t)^{1-z} \\
p(z \mid r_t) &= \int_0^1 p(z \mid q_t) p(q_t = u \mid r_t) \, du \\
&= \int_0^1 u^z \, (1-u)^{1-z} \, p(q_t = u \mid r_t) \, du \\
&= \begin{cases} \int_0^1 (1-u) \, p(q_t = u \mid r_t) \, du, & z = 0 \\ \int_0^1 u \, p(q_t = u \mid r_t) \, du, & z = 1 \end{cases} \\
&= \begin{cases} 1 - \mathbf{E}[R_t \mid r_t], & z = 0 \\ \mathbf{E}[R_t \mid r_t], & z = 1 \end{cases} \\
&= r_t^z \, (1 - r_t)^{1-z}
\end{aligned}
\tag{2.13}
$$

since by definition $r_t = \mathbf{E}[R_t \mid r_t]$. Fig. 2.1 shows the corresponding dependency graph of our model.

Step 9. Deciding whether to stop

We may now use the distribution over the models to predict the expected error should we choose to add k more examples. This is simply

$$\mathbf{E}[R_{t+k} \mid \xi_t] = \int_\Omega g(t+k;\omega) p(\omega \mid \xi_t)\, d\omega, \tag{2.14}$$

where the calculation required for step 9 of Alg. 5 follows trivially.

Specifics of the model

What remains unspecified is the set of convergence curves $h(t; c, \varrho)$ that will be employed. We shall make use of curves related to common theoretical convergence results. It is worthwhile to keep in mind that we simply aim to find the combination of the available estimates that gives the best predictions. While none of the estimates might be particularly accurate, we expect to obtain reasonable stopping times when they are optimally combined in the manner described in the previous section (Eq. 2.6). Ultimately, we expect to end up with a fairly narrow distribution over the possible convergence curves.

One of the weakest convergence results [36] is for sample complexity of order $\mathcal{O}(1/\epsilon_t^2)$, which corresponds to the *quadratic* convergence curve

$$h(t; 3, \kappa) = \sqrt{\frac{\kappa}{t+\kappa}}, \quad \kappa > 0, \tag{2.15}$$

and is denoted using $c = 3, \varrho = \kappa$. Another common type is for sample complexity of order $\mathcal{O}(1/\epsilon_t)$, which corresponds to the *linear* convergence curve

$$h(t;\, 2, \lambda) = \frac{\lambda}{t+\lambda}, \quad \lambda > 0. \tag{2.16}$$

A final possibility is that the error decreases exponentially fast. This is theoretically possible in some cases (see Sec. 2.1 for an example), as was proved in [6]. The resulting sample complexity of order $\mathcal{O}(\log(1/\epsilon_t))$ corresponds to the *exponential* convergence curve

$$h(t; 1, \beta) = 2^{-t/\beta}, \quad \beta > 0. \tag{2.17}$$

As a simple illustration, we examined the performance of the estimation and the stopping criterion in a simple classification problem with data of 10 classes, each with an equivariant Gaussian distribution in an 8-dimensional space. Each new example was simply classified as having the label closest to the empirical mean of the observations for each class. All examples were chosen randomly, i.e. $\mathcal{D}_A = \emptyset$.

As can be seen in Fig. 2.2, at the initial stages the estimates are inaccurate. This is due to two facts: (a) The distribution over convergence rates

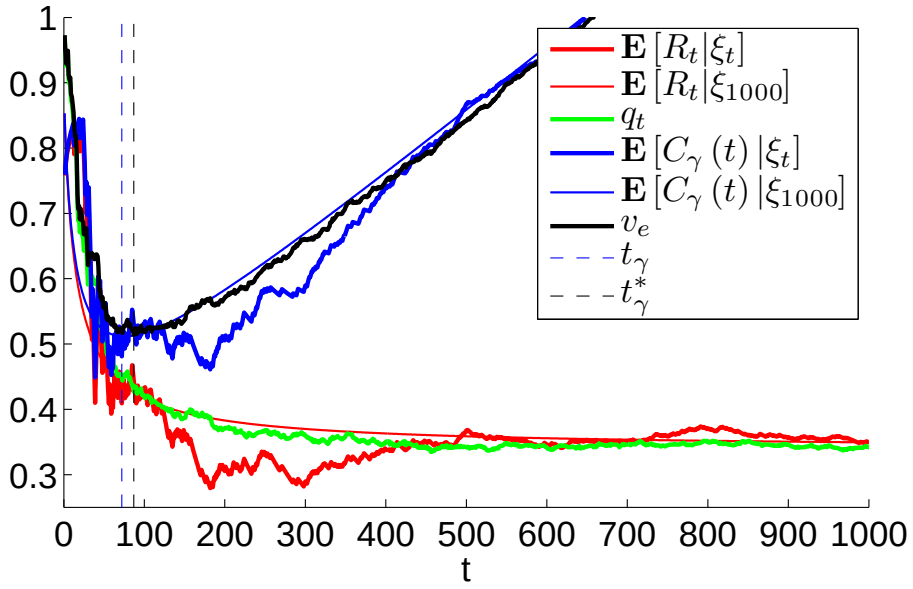


Figure 2.2: Illustration of the estimated error on a 10-class problem with a cost per label of $\gamma = 0.001$. On the vertical axis, is the **history** of the predicted generalisation error and the predicted costs, i.e $\mathbf{E}[R_t \mid \xi_t]$ and $E[C_\gamma(t) \mid \xi_t]$, while q_t is the **generalisation error** a measured on a test-set of size 1000 and v_e is the corresponding actual **cost**. Finally, $\mathbf{E}[R_t \mid \xi_{1000}]$ and $E[C_\gamma(t) \mid \xi_{1000}]$ are the final **estimated** learning and cost *curves* given all the observations. The stopping time t_γ of OBSV is indicated by the blue dashed line, while the the optimal stopping time t^*_γ indicated by the black dashed line.

is initially dominated by the prior. As more data is accumulated, there is better evidence for what the final error will be. (b) As we mentioned in the discussion of step 7, the realised convergence curve is much more random than the expected convergence curve which is actually modelled. However, as the number of examples approaches infinity, the expected and realised errors converge. The stopping time t_γ for OBSV is nevertheless relatively close to the optimal stopping time t^*_γ, as v_e appears to be minimised near 90. In the following chapter we will present our analytically justified choice for the unknown convergence parameters and an extensive evaluation of this stopping algorithm.

Chapter 3

Experimental evaluation of OBSV

The purpose of this chapter is to evaluate OBSV. First we describe our choice of parameters for the learning curves given the *difference in cost* as measure of performance (Sec. 3.1). Then we evaluate OBSV using artificial observations that arise from those learning curves (Sec. 3.2). Using those we evaluate the various predictions by our model. To evaluate the whole model we follow [8] and plot performance curves for a range of values of γ. Thereby we relate the performance to different baselines. Then we move to real data sets (Sec. 3.3). There we also evaluate different combinations of sampling strategies and learning algorithms (Sec. 3.4). Finally we evaluate the performance of the combination of the image categorisation framework from Chap. 1 and OBSV. For this purpose we utilise image data from `VOC06` and show how to chose γ given two scenarios from practice.

3.1 Parameter selection

We want to evaluate OBSV by the difference in cost when compared to another stopping algorithm. For this we analyse the dependency of this performance measure on the convergence parameters. As a result we will obtain a set convergence models (Sec. 3.1.1). Finally we will restrict these models to those that are reasonable for practice (Sec. 3.1.2).

3.1.1 Solutions towards a constant increase in cost

As shown in Sec. 2.2.2 we model the expected cost at time t given a learning algorithm F and a data distribution $\mathscr{D}$ by some $\omega \in \Omega$, i.e.

$$\begin{aligned}\mathbf{E}[C_\gamma(t) \mid F, \mathscr{D}] &= \mathbf{E}\left[R_t \mid F, \mathscr{D}\right] + \gamma t = r_t + \gamma t \\ &= g(t;\omega) + \gamma t = r_\infty + (r_0 - r_\infty) \cdot h(t; c, \varrho) + \gamma t.\end{aligned} \tag{3.1}$$

Then the optimal stopping time $t_\gamma(\omega)$ that minimises the expected costs given the three classes of convergence curves according to Eq. 2.15 - 2.17 is

$$t_\gamma(1, \beta, r_0, r_\infty) = \frac{\beta \cdot \ln\left(\frac{\ln(2)(r_0 - r_\infty)}{\gamma\beta}\right)}{\ln(2)}, \quad 0 < \beta < \frac{\ln(2)(r_0 - r_\infty)}{\gamma} \tag{3.2}$$

$$t_\gamma(2, \lambda, r_0, r_\infty) = \sqrt{\frac{\lambda\,(r_0 - r_\infty)}{\gamma}} - \lambda, \quad 0 < \lambda < \frac{r_0 - r_\infty}{\gamma} \tag{3.3}$$

$$t_\gamma(3, \kappa, r_0, r_\infty) = \kappa \left(\sqrt[3]{\left(\frac{r_0 - r_\infty}{2\gamma\kappa}\right)^2} - 1 \right), \quad 0 < \kappa < \frac{r_0 - r_\infty}{2\gamma} \tag{3.4}$$

where the ranges given on the right-hand side ensure $t_\gamma(\omega) > 0$ and we further note that the optimal stopping time itself is maximised for

$$\beta_0 = \frac{\ln(2)(r_0 - r_\infty)}{e\gamma} \tag{3.5}$$

$$\lambda_0 = \frac{r_0 - r_\infty}{4\gamma} \tag{3.6}$$

$$\kappa_0 = \frac{\sqrt{3}\,(r_0 - r_\infty)}{18\gamma}. \tag{3.7}$$

Assume we are given a sorted set of convergence parameters ϱ. According to Eq. 3.1, the difference in cost inferred by OBSV for stopping according to a wrong estimate ϱ_{i+1} instead of ϱ_i is

$$\begin{aligned}\Delta_\gamma(\varrho_{i+1}, \varrho_i; c, r_0, r_\infty) = \\ (r_0 - r_\infty)\,[h\,(t_\gamma(c, \varrho_{i+1}, r_0, r_\infty); c, \varrho_i) \;-\; h\,(t_\gamma(c, \varrho_i, r_0, r_\infty); c, \varrho_i)] \\ + \gamma\,[t_\gamma(c, \varrho_{i+1}, r_0, r_\infty) \;-\; t_\gamma(c, \varrho_i, r_0, r_\infty)]\,.\end{aligned} \tag{3.8}$$

For the purpose of evaluation we want Δ_γ to remain constant over the whole range of possible convergence prameters ϱ. That is, for any two consecutive convergence parameters ϱ_{i+1}, ϱ_i we require that

$$\Delta_\gamma(\varrho_{i+1}, \varrho_i; c, r_0, r_\infty) = \Delta_\gamma(\varrho_i, \varrho_{i+1}; c, r_0, r_\infty) = K\,(r_0 - r_\infty) \tag{3.9}$$

for some constant value $K > 0$ at scale $(r_0 - r_\infty)$.

Letting ϵ denote the difference between two consecutive convergence parameters, i.e.

$$\varrho_{i+1} = \varrho_i + \epsilon, \tag{3.10}$$

we reformulate Eq. 3.9 to

$$\Delta_\gamma(\varrho + \epsilon, \varrho; c, r_0, r_\infty) = \Delta_\gamma(\varrho, \varrho + \epsilon; c, r_0, r_\infty) = K(r_0 - r_\infty) \tag{3.11}$$

Thus, given ϱ_0 and the solution of Eq. 3.11 for ϵ, we could simply calculate all other convergence parameters according to Eq. 3.10. Unfortunately a

symbolic solution to Eq. 3.11 is hard to obtain. Therefore we consider the two subequations

$$\Delta_\gamma(\varrho+\epsilon,\varrho;c,r_0,r_\infty)-K(r_0-r_\infty)=0 \tag{3.12a}$$
$$\Delta_\gamma(\varrho,\varrho+\epsilon;c,r_0,r_\infty)-K(r_0-r_\infty)=0 \tag{3.12b}$$

of Eq. 3.11 instead and show that the solution of Eq. 3.12a for ϵ yields a good approximation for solution of Eq. 3.12b.

By substituting

$$\varrho=\varrho'(r_0-r_\infty)/\gamma \tag{3.13a}$$
$$\epsilon=\epsilon'(r_0-r_\infty)/\gamma \tag{3.13b}$$

in Eq. 3.12a, the dependency on r_0, r_∞ and γ vanishes and we obtain the following general equations given the three classes of convergence,

$$K=2^{-\frac{(\beta'+\epsilon')}{\beta'\ln(2)}\ln\left(\frac{\ln(2)}{\beta'+\epsilon'}\right)}-2^{-\ln\left(\frac{\ln(2)}{\beta'}\right)}+\frac{(\beta'+\epsilon')\cdot\ln\left(\frac{\ln(2)}{\beta'+\epsilon'}\right)}{\ln(2)}-\frac{\beta'\cdot\ln\left(\frac{\ln(2)}{\beta'}\right)}{\ln(2)} \tag{3.14}$$

$$K=\frac{\lambda'}{\sqrt{\lambda'+\epsilon'}-\epsilon'}-2\sqrt{\lambda'}-\epsilon'+\sqrt{\lambda'+\epsilon'} \tag{3.15}$$

$$K=\sqrt{\frac{2\kappa'}{\sqrt[3]{2(\kappa'+\epsilon')}-2\epsilon'}}-\frac{3}{2}\sqrt[3]{2\kappa}-\epsilon'+\sqrt[3]{2(\kappa'+\epsilon')}. \tag{3.16}$$

The solutions to Eq. 3.14 are given by

$$\epsilon'=-\beta'+\frac{\beta'\left(-W\left(b_1,-\exp\left(-\frac{\beta'+K\ln(2)}{\beta'}\right)\right)-\frac{\beta'+K\ln(2)}{\beta'}+\ln\left(\frac{\beta'}{\ln(2)}\right)\right)\right)}{W\left(b_2,\beta'\left(-W\left(b_1,-\exp\left(-\frac{\beta'+K\ln(2)}{\beta'}\right)\right)-\frac{\beta'+K\ln(2)}{\beta'}+\ln\left(\frac{\beta'}{\ln(2)}\right)\right)\right)\right)}, \tag{3.17}$$

where W denotes the Lambert W-function with its branches $b_{1,2}\in\{-1,0,1\}$, and the solutions of Eq. 3.15 are given by

$$\epsilon'=-\lambda'+\left(\frac{1+\sqrt{(1-2\sqrt{\lambda'})^2-2K\pm2\sqrt{K^2+4K\sqrt{\lambda'}}}}{2}\right)^2. \tag{3.18}$$

Unfortunately the solutions of Eq. 3.16 have no concise form and since this causes serious numerical problems, we approximate ϵ' numerically. Fig. 3.1(a) shows the actual values of ϵ' as function of β' for $K=4.125\cdot10^{-4}$ according to the two branches ($b_1=-1, b_2=-1$) and ($b_1=-1, b_2=0$) of the Lambert W-function. Thus starting at $\beta'_0=\ln(2)/e$ we use the upper branch

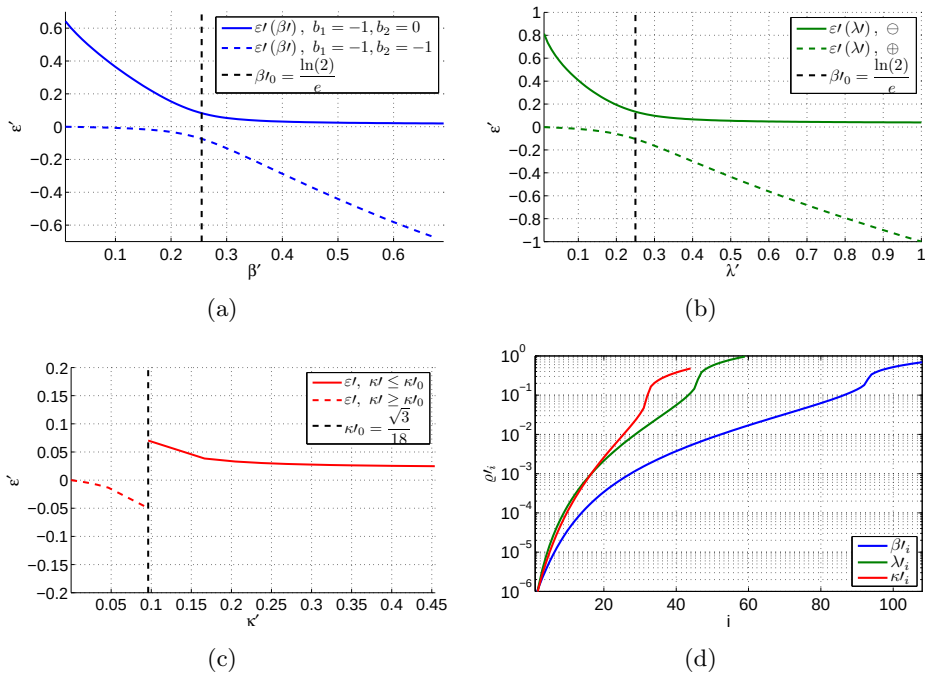


Figure 3.1: The solutions for ϵ' as a function of ϱ' and the resulting parameters for $K = 4.125 \cdot 10^{-4}$. Fig. 3.1(a) shows the two solutions given exponential parameters according to the branches $(b_1 = -1, b_2 = -1)$ and $(b_1 = -1, b_2 = 0)$ of the Lambert W-function. Fig. 3.1(b) shows the two solutions given linear parameters. Fig. 3.1(c) shows the numerical solution given quadratic parameters. The black dashed lines show ϱ'_0 for each class of convergence. Fig. 3.1(d) shows the resulting $\varrho' \geq 10^{-7}$.

to calculate all $\beta'_0 < \beta'_i < \log(2)$ and the lower branch for all $0 < \beta'_i < \beta'_0$. Although the choice of K is essentially arbitrary, we chose K in order to obtain about 50 values for κ'. As can be seen by Fig. 3.1(b) - 3.1(c) the ϵ'-curves for linear and quadratic models look similar. Fig. 3.1(d) shows the resulting general convergence parameters ϱ'. For the experiments we will determine all ϱ from ϱ' through back-substitution given the different values of γ, r_0, r_∞ used.

We now validate the resulting differences in cost according to Eq. 3.12a - 3.12b. Fig. 3.2 shows the differences in cost for the three classes of convergence models when estimating $\varrho_{i\pm1}$ instead of the true convergence parameter ϱ_i given for example $\gamma = 10^{-6}, r_0 = 0.5, r_\infty = 0$ and $K = 4.125 \cdot 10^{-4}$. As can be seen there the difference in cost is reasonably close to $K \cdot (r_0 - r_\infty)$. Only for exponential models when estimating the β_{i+1} instead of β_i the difference in cost is lower than $K \cdot (r_0 - r_\infty)$ by a bit more than an order of magnitude. However, for larger values of γ or r_∞ this gap becomes less

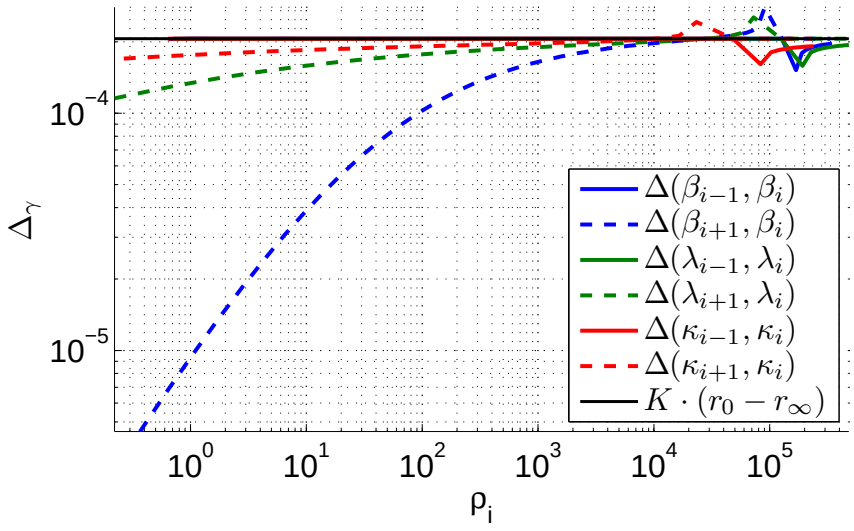


Figure 3.2: The difference in cost for the three classes of convergence models when estimating $\varrho_{i\pm1}$ instead of the true convergence parameter ϱ_i given $\gamma = 10^{-6}, r_0 = 0.5, r_\infty = 0$ and $K = 4.125 \cdot 10^{-4}$.

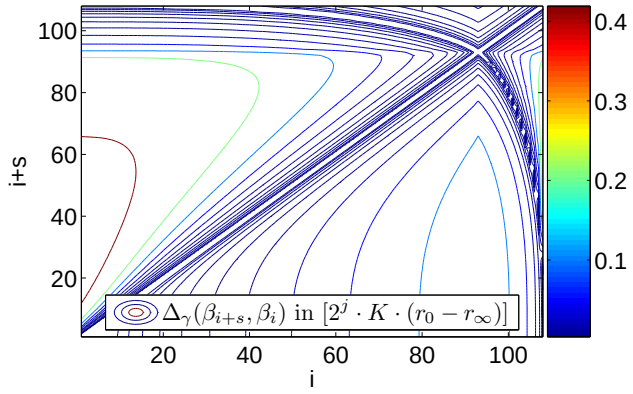
than an order of magnitude. Its worthwhile mentioning that we also tried to balance the differences in cost by solving

$$\epsilon = \arg\min_{\epsilon'} \left(\Delta(\varrho'_{i+1}, \varrho'_i, c, r_0, r_\infty) - K\right)^2 - \left(\Delta(\varrho'_i, \varrho'_{i+1}, c, r_0, r_\infty) - K\right)^2 \quad (3.19)$$

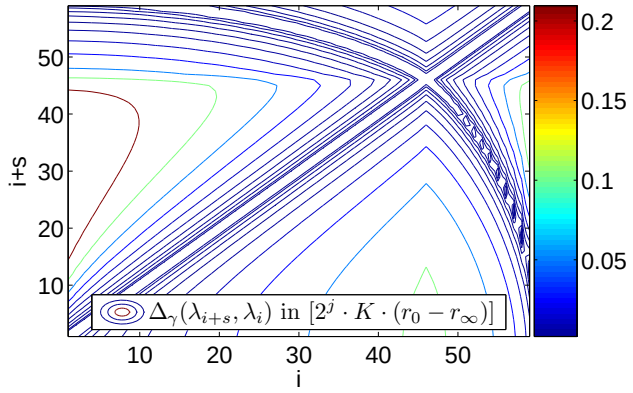
instead of Eq. 3.12a. Since it is hard to obtain a symbolic solution for Eq. 3.19, we calculated a numeric solution. This however yielded almost identical differences in cost (Eq. 3.8) as those obtained by the solution of Eq. 3.12a. Also note that we observed such a similarity of the solutions for all valid offsets $s \in \mathbb{Z}$ from i. Therefore we argue that our choice of the parameters is close to optimal with respect to Eq. 3.19. It is clear that with increasing offset $|s|$ the difference in cost will deviate more from $|s| \cdot K(r_0 - r_\infty)$. However as shown by Fig. 3.3 our performance measure is reasonably linear in the neighbourhood of the true parameter ϱ. As can be seen there the contours are x-shaped. The reason is that, since ϱ_0 is maximising the optimal stopping time, there are always some ϱ_i, ϱ_j such that $\varrho_i < \varrho_0 < \varrho_j$ and $t_\gamma(c, \varrho_i, r_0, r_\infty) \approx t_\gamma(c, \varrho_j, r_0, r_\infty) < t_\gamma(c, \varrho_0, r_0, r_\infty)$.

3.1.2 Convergence models for practice

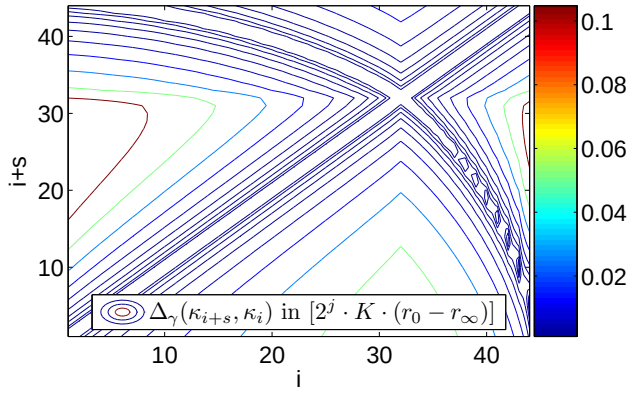
Finally we restrict our parameters such that the remaining models do not converge faster than $h(t; 1, 1) = 2^{-t}$, which is the fastest convergence currently known from practice (Sec. 2.1). That is, we want to use only those λ



(a)



(b)



(c)

Figure 3.3: The difference in costs for the three classes of convergence models when estimating ϱ_{i+s} instead of the true convergence parameter ϱ_i given valid $s \in \mathbb{Z}$ and $\gamma = 10^{-6}, r_0 = 0.5, r_\infty = 0$ and $K = 4.125 \cdot 10^{-4}$. The levels of the contours shown correspond to $2^j \cdot K \cdot (r_0 - r_\infty),\ j = 1, 2, \ldots$.

and κ for which

$$2^{-t} \leq 2^{-t/\beta} \tag{3.20}$$

$$2^{-t} \leq \frac{\lambda}{\lambda + t} \tag{3.21}$$

$$2^{-t} \leq \sqrt{\frac{\kappa}{\kappa + t}} \tag{3.22}$$

for all $t \geq 0$. Solving for β, λ of κ respectively, we obtain

$$\beta \geq 1$$

$$\lambda \geq \frac{t \cdot 2^{-t}}{1 - 2^{-t}} \tag{3.23}$$

$$\kappa \geq \frac{t \cdot 2^{-2t}}{1 - 2^{-2t}} \tag{3.24}$$

and by maximising the right-hand sides of Eq. 3.23 - 3.24 over all $t \geq 0$ we obtain the lower bounds for our convergence parameters

$$\beta \geq 1 \tag{3.25}$$

$$\lambda \geq \frac{1}{\ln(2)} \tag{3.26}$$

$$\kappa \geq \frac{1}{2\ln(2)}. \tag{3.27}$$

We will refer to such as *admissible* convergence parameters. Fig. 3.4 - 3.5 show examples of admissible convergence parameters according to Eq. 3.13a as functions of $r_\infty \in \mathcal{R}_\infty$ (left column) and $\gamma \in \Gamma$ (right column), where we chose

$$\begin{aligned} \mathcal{R}_\infty &= \{0, 1/300, 2/300, \ldots, 149/300\}, \\ \Gamma &= \{9 \cdot 10^{-k}, 8 \cdot 10^{-k}, \ldots, 1 \cdot 10^{-k}\},\ k = 1, \ldots, 6, \end{aligned}$$

and $r_0 = 0.5$, which we will also use for the experiments. Furthermore we will denote the set of models ω that result from all admissible convergence parameters ϱ_i given some specific values for γ, c, r_0, r_∞ by $\Omega_{\gamma,c,r_0,r_\infty}$. Note that we will slightly abuse notation and use shorthands of the form

$$\Omega_{\gamma,c,r_0} = \bigcup_{r_\infty \in \mathcal{R}_\infty} \Omega_{\gamma,c,r_0,r_\infty}.^{1}$$

Furthermore will use $m_\gamma(c, r_0, r_\infty) = |\Omega_{\gamma,c,r_0,r_\infty}|$ to denote the total number of admissible convergence parameters ϱ_i given specific values for γ, c, r_0, r_∞. Then for example $m_{10^{-6}}(1, 0.5, 0) = 106$, $m_{10^{-6}}(2, 0.5, 0) = 57$ and $m_{10^{-6}}(3, 0.5, 0) = 42$.

[1] This means that a missing subscript indicates the union of the sets indexed by it.

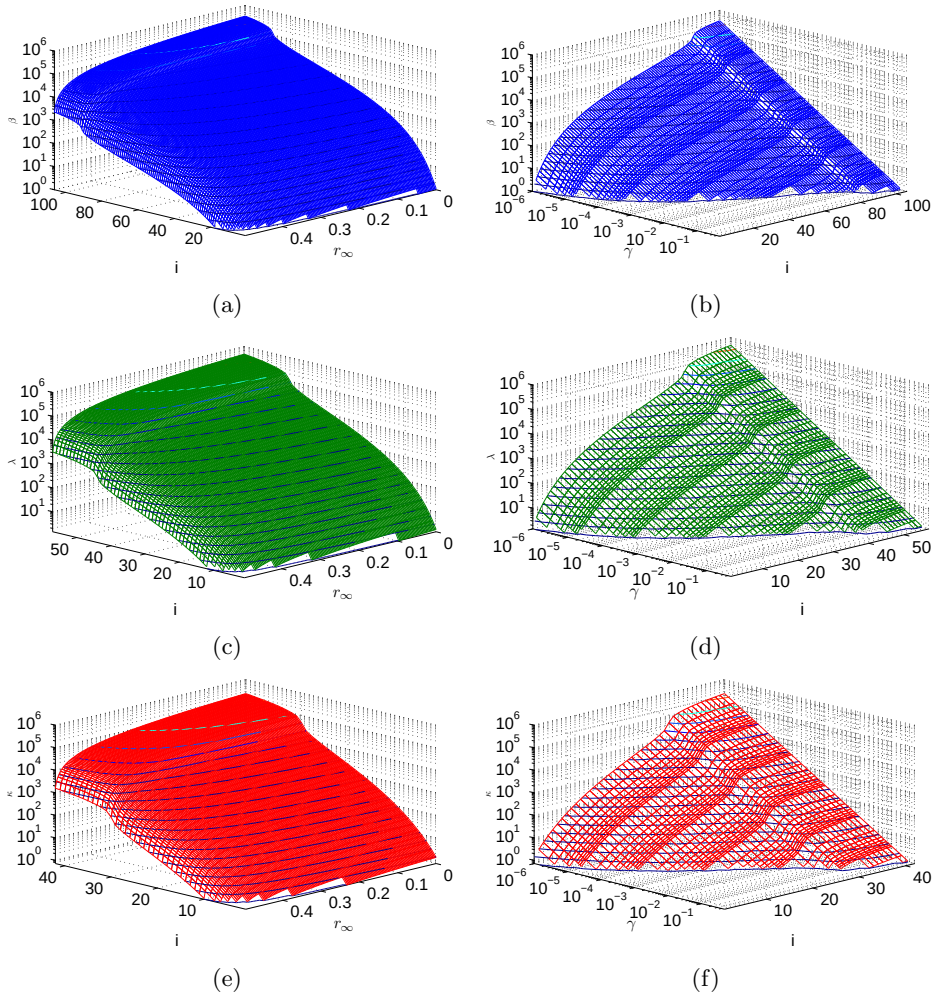


Figure 3.4: Some admissible convergence parameters given the three different classes of convergence. The left column shows the convergence parameters given $r_0 = 0.5$ and $\gamma = 10^{-6}$ as a function of r_∞, while the right column shows the convergence parameters given $r_0 = 0.5, r_\infty = 0$ as a function of γ. The black horizontal lines correspond to multiples of the lower bound for the convergence parameters as given by Eq. 3.25 - 3.27. More specifically, a black line corresponds to a level of 2^j in case of the exponential parameters (Fig. 3.4(a) - 3.4(b)), of $2^j/\ln(2)$ in case of linear parameters (Fig. 3.4(c) - 3.4(d)) and of $2^j/(2\ln(2))$ in case of quadratic parameters (Fig. 3.4(e) - 3.4(f)) with $j = 0, 1, 2, \ldots$.

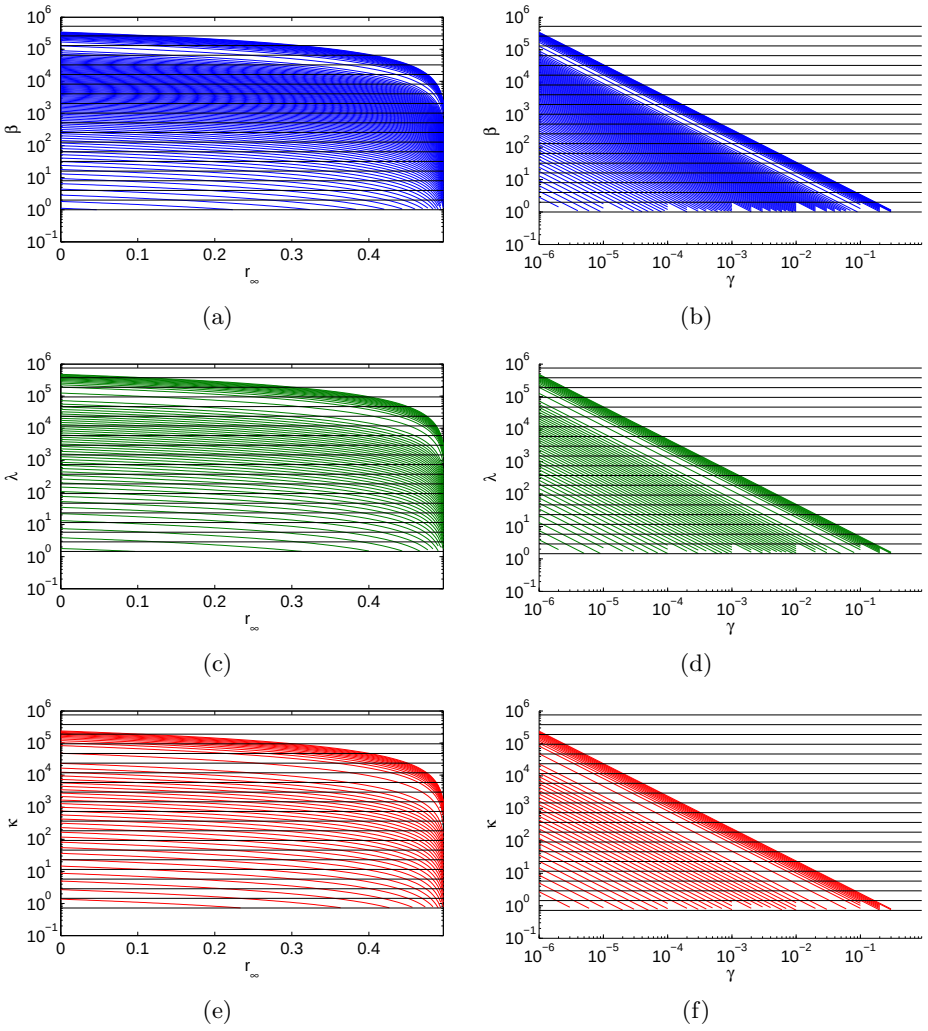


Figure 3.5: Projections of the admissible convergence parameters shown in Fig. 3.4. The left column shows the convergence parameters given $r_0 = 0.5$ and $\gamma = 10^{-6}$ as a function of r_∞ projected on the ϱ, r_∞-plane. The right column shows the convergence parameters given $r_0 = 0.5, r_\infty = 0$ as a function of γ projected on the ϱ, γ-plane. The black horizontal lines correspond to multiples of the lower bound for the convergence parameters as given by Eq. 3.25 - 3.27. More specifically, a black line corresponds to a level of 2^j in case of the exponential parameters (Fig. 3.4(a) - 3.4(b)), of $2^j / \ln(2)$ in case of linear parameters (Fig. 3.4(c) - 3.4(d)) and of $2^j/(2\ln(2))$ in case of quadratic parameters (Fig. 3.4(e) - 3.4(f)) with $j = 0, 1, 2, \ldots$.

Choosing the prior for the models

As can be seen by Fig. 3.4(a), 3.4(c), 3.4(e) the grid of convergence parameters over the pairs (i, r_∞) is not fully populated. More specifically, there are no entries for $i \to 1$ and $r_\infty \to 0.5^-$ since these would correspond to non-admissible convergence parameters which violate Eq. 3.25 - 3.27. Therefore, as $r_0 = 0.5$ is given, it is reasonable to model the distribution of the unknown parameters ϱ and r_∞ as a joint distribution for each class of convergence,

$$\mathbf{P}(\omega \mid \xi_t) = \mathbf{P}(\omega \mid c, \xi_t)\, \mathbf{P}(c \mid \xi_t) = \mathbf{P}(\varrho, r_\infty \mid c, \xi_t)\, \mathbf{P}(c \mid \xi_t), \tag{3.28}$$

since $\Omega_{\gamma,c} \cap \Omega_{\gamma,c'} = \emptyset$ for $c \neq c'$. Since we do not know which class of convergence is more probable a priori, we will choose the prior by distributing the probability mass uniformly over all convergence classes and then also uniformly amongst all models that belong to each class,

$$\mathbf{P}(\omega \mid \xi_0) = \frac{1}{\sum\limits_{r_\infty \in \mathcal{R}_\infty} m_\gamma\left(c = j, 0.5, r_\infty\right)} \cdot \frac{1}{3}. \tag{3.29}$$

A lower bound on γ

As can be seen by Fig. 3.4(b), 3.4(d), 3.4(f) and Fig. 3.5(b), 3.5(d), 3.5(f), $m_\gamma\left(c, r_0, r_\infty\right)$ becomes smaller or even zero as γ increases. More specifically, by Eq. 3.25 - 3.27 and Eq. 3.2 - 3.4 is follows that

$$1 \leq \frac{\ln(2)\,(r_0 - r_\infty)}{\gamma} \Rightarrow \gamma \leq r_0 \ln(2), \tag{3.30}$$

since $r_\infty \geq 0$. Thus, when $\gamma > r_0 \ln(2)$, there is no admissible convergence parameter, and therefore no admissible model ω, under the assumptions made and we should stop immediately without querying any labels at all.

However, the upper bound on γ for immediate stopping is even lower. Specifically the reduction in error is at most

$$g(t; 1, 1, r_0, r_\infty) - g(t+1; 1, 1, r_0, r_\infty) \geq g(t; \omega) - g(t+1; \omega),$$

$\forall \omega \in \Omega$ and maximised at $t = 0$. Therefore OBSV should query any label only when

$$\gamma \leq g(0; 1, 1, r_0, r_\infty) - g(1; 1, 1, r_0, r_\infty) = \frac{1}{2}\,(r_0 - r_\infty) \leq \frac{r_0}{2}, \tag{3.31}$$

since $r_\infty \geq 0$. Thus for $\gamma \to \left(\frac{r_0}{2}\right)^-$ the set of admissible convergence models becomes smaller and therefore biased towards exponential convergence, i.e. $h(t, 1, 1)$. For such values of γ we cannot model $h(t), r_\infty$ or r_t when the actual convergence is slower. However, in such cases the optimal stopping time is zero and since our main focus is optimal stopping this point is irrelevant.

For reference, Fig. 3.6 shows some convergence and cost curves given $\gamma = 2 \cdot 10^{-4}$ and $r_\infty = 0$.

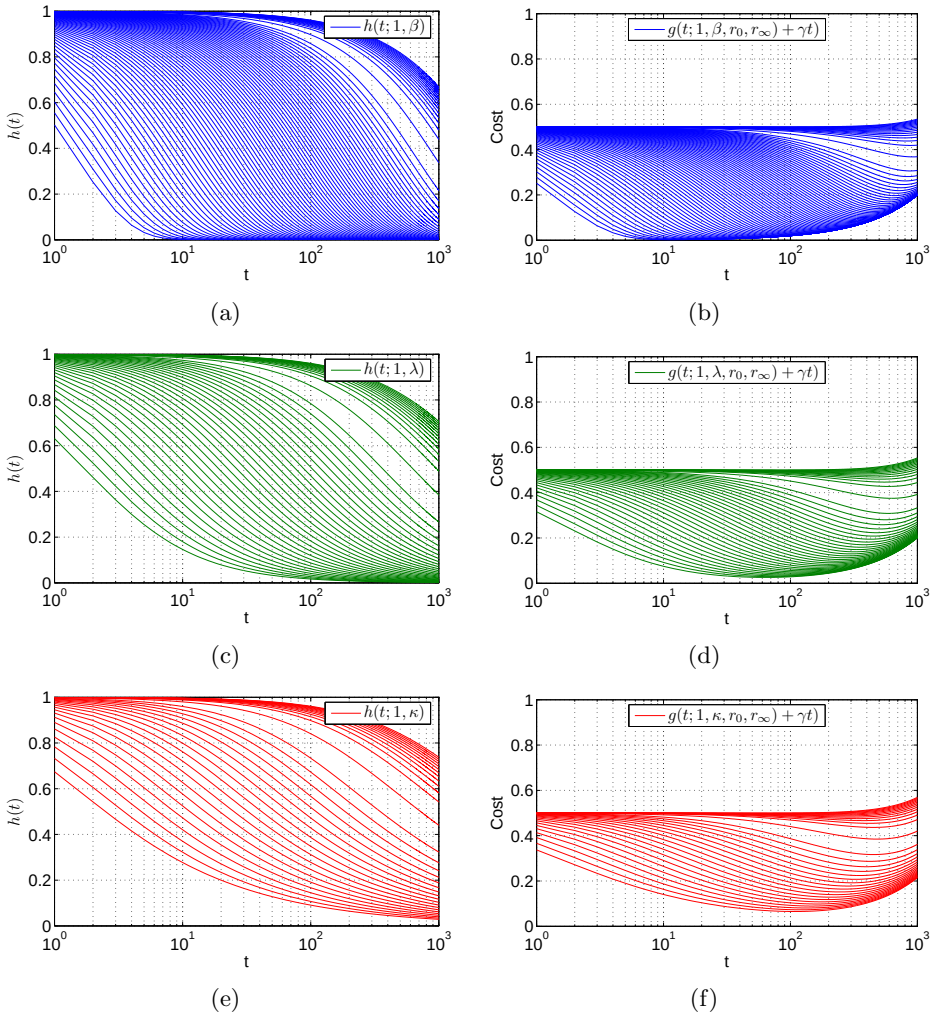


Figure 3.6: The final convergence and cost curves for the experiments given the three different classes of convergence at $\gamma = 2{\cdot}10^{-4}, r_0 = 0.5$ and $r_\infty = 0$.

3.2 Artificial observations

It is laborious to choose combinations of datasets and learning algorithms such that the resulting learning curves would cover those one can generally expect to observe in practice (Sec. 3.1.2). Therefore we choose to run a controlled experiment where we generate artificial observations $\mathbf{z}_t$ by sampling along learning curves that stem from Ω_γ. Using such observations we evaluate the various predictions of our model for different values of γ. These values of γ are also used as input to the stopping algorithm. We evaluate OBSV using the resulting stopping times and costs. Finally we relate the performance of our model to maximum a posteriori predictions and a naive stopping criterion.

Let ξ^* be the *true belief* over Ω_γ. At every time step t we generate m_t random observations $\mathbf{z}_t = (z_t(1), \ldots, z_t(m_t)) \in \{0,1\}^{m_t}$ of the error, where

$$\mathbf{P}[z_t(i) = 1 \mid \xi^*] = r_t \equiv \mathbf{E}[R_t \mid \xi^*], \quad i = 1, \ldots, m_t,$$

calculate their mean v_t and update our belief ξ_{t-1} according to Eq. 2.10 - 2.12. Thus for each ξ^* we can generate a sequence of artificial observations $\langle \mathbf{z}_t \rangle = (\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_m)$ along the corresponding artificial learning curve.

For our experiments we choose various ξ^*, each corresponding to a *true model* $\omega^* \in \Omega^*_\gamma \subset \Omega_\gamma$, i.e. $\mathbf{P}(\omega^* \mid \xi^*) = 1$. More specifically, given $r^*_0 = 0.5$ and some value of γ, for each c^* we choose various $\omega^* = (c^*, \varrho^*, r^*_0, r^*_\infty)$ using a grid over ϱ and r_∞. That is, letting

$$r^*_\infty \in \mathcal{R}^*_\infty = \{0, 0.05, 0.1, \ldots, 0.45\} \subset \mathcal{R}_\infty$$

we select

$$\Omega^*_\gamma = \bigcup_{c^*=1,2,3} \Omega_{\gamma,c^*} \cap \bigcup_{r^*_\infty \in \mathcal{R}^*_\infty} \left\{ (c^*, \varrho^*_i, 0.5, r^*_\infty) \mid i = \left\lceil 0.5 + k \cdot \frac{m_0(c^*, 0.5, r^*_\infty) - 1}{9} \right\rceil \text{ with } k = 0, \ldots, 9 \right\}. \tag{3.32}$$

For example, given $\gamma = 10^{-6}$ we obtain our exponential targets $\Omega^*_{\gamma,c^*=1}$ by intersecting the models shown in Fig. 3.4(a) with those on the grid. It is important to note that we choose the prior for the targets by

$$\mathbf{P}\left(\omega^* \mid \Omega^*_\gamma\right) = \frac{1}{|\Omega^*_\gamma|}. \tag{3.33}$$

and that, as $\gamma < r_0/2$ within the experiments, $m_\gamma(c, 0.5, r_\infty) > 0$ and therefore Ω_γ will always include the largest admissible convergence parameter according to the ranges given on the right side of Eq. 3.2 - 3.4.

For each target ω^* we perform 10 randomised runs making $m = 1000$ observations of the true error, i.e. $m_t = 1$ and $t = 1, \ldots, m$. To keep the optimal stopping times $t_\gamma(\omega) < m$ for all $\omega \in \Omega_\gamma$, we will only consider query costs of $\gamma \geq 2 \cdot 10^{-4}$. This gives a total number of 34310 runs.

Thereby we will evaluate the *prediction of the expected convergence* and *of the expected final error given* ξ_t

$$\mathbf{E}[h(t; c, \varrho) \mid \xi_t] = \sum_{\omega \in \Omega_\gamma} h(t; c, \varrho)\, \mathbf{P}(\omega \mid \xi_t) \tag{3.34a}$$

$$\mathbf{E}[r_\infty \mid \xi_t] = \sum_{\omega \in \Omega_\gamma} r_\infty\, \mathbf{P}(\omega \mid \xi_t), \tag{3.34b}$$

as well as the prediction of the expected error according to Eq. 2.14. It is important to note that the prior of the targets as given by Eq. 3.33 is uniform whereas the prior of the models as given by Eq. 3.29 depends on the convergence class. Therefore we utilise importance sampling. More specifically, when averaging the results over *different* classes of target convergence, we will *weight* the result for each target $\omega^* \in \Omega^*_\gamma$ by its (normalised) *probability given the prior belief* ξ_0 of our model,

$$\mathbf{P}\left(\omega^* \mid \Omega^*_\gamma, \xi_0\right) = \frac{\mathbf{P}\left(\omega^* \mid \xi_0\right)}{\sum\limits_{\tilde{\omega}^* \in \Omega^*_\gamma} \mathbf{P}\left(\tilde{\omega}^* \mid \xi_0\right)}, \tag{3.35}$$

where $\mathbf{P}\left(\omega^* \mid \Omega^*_\gamma, \xi_0\right) \neq \mathbf{P}\left(\tilde{\omega}^* \mid \Omega^*_\gamma, \xi_0\right)$ only if $c^* \neq \tilde{c}^*$.

Since OBSV is a deterministic stopping algorithm we obtain one stopping time $t_\gamma \leftarrow \mathrm{OBSV}(\gamma)$ for each run and value of γ. Thus we also calculate the (weighted) averages in t_γ and in $v_e(\gamma, t_\gamma, \langle \mathbf{z}_t \rangle)$ according to Eq. 2.2 over all runs. By examining those averages and their extreme values we are able to estimate the sensitivity of our results to the target belief.

3.2.1 Prediction of the expected convergence and the expected final error

First we compare the prediction of the expected convergence to its true value $h(t; c^*, \varrho^*)$. Fig. 3.7 (left column) shows the weighted average in absolute difference between the predicted convergence and the target value as a function of t given different values of γ. As can be seen the average reaches a maximum after approximately 10 to 20 observations and converges towards zero afterwards. However for increasing values of γ, OBSV needs an decreasing number of observations to approach the true convergence to a certain level, i.e. the progress is faster. The reasons are that (a) $|\Omega_\gamma|$ becomes smaller and (b) $\omega^* \in \Omega^*_\gamma$ become biased towards faster convergence.

Next we investigate the sensitivity of these results given different classes of target convergence. Although these results do not differ significantly, they

provide useful insights. As shown by Fig. 3.7 (right column) when t is small the convergence results for exponential targets ($c^* = 1$) are a bit worse compared to linear or quadratic ones ($c^* = 2, 3$). Later on the situation changes and our model performs worst when $c^* = 3$. The rationale behind this is (a) the model is initially dominated by the prior belief ξ_0 and the uncertainty about a particular observation v_t of r_t, which (b) could have also been sampled from a learning curve of slower convergence but smaller r_∞ or (c) from a learning curve of faster convergence but larger r_∞. As the the prior belief represents something between exponential and quadratic convergence our model favours linear targets. Thus the absolute difference in convergence is affected the most when $c^* = 1$ and t is small, since many of those targets converge very fast (see Fig. 3.6(a) for an example of such convergence curves). However with an increasing number of observations the uncertainty becomes smaller and the absolute difference in convergence goes to zero. A similar argument holds for the comparably worse performance of our model when $c^* = 3$ and t is large. There we find that some of the exponential and linear models are still assigned some probability and in turn the performance upon quadratic targets is worse compared to linear ones. Vice versa for slow exponential targets at late points in time, there is still some probability for linear or quadratic alternatives which can be seen by the $95th$-percentiles for $c^* = 1$ at $\gamma = 2 \cdot 10^{-3}, 2 \cdot 10^{-4}$.

Next we validate these findings by investigating the predicted final error and its true value r^*_∞. As shown by Fig. 3.8 (left column) the weighted average in their absolute difference becomes smaller with increasing t. We note that corresponding to the results above the progress made becomes faster as γ increases. Although the results given different classes of target convergence (Fig. 3.8, left column) do not differ significantly they provide useful insight. More specifically, for each value of γ both the average and the $95th$-percentile given $c^* = 3$ are always a bit larger than in case of $c^* = 1, 2$. This is due to the fact that, given $c^* = 3$, at later points in time the uncertainty about the target is comparably larger than in case of other c^* since the latter converge earlier. More precisely, for such t where the remaining alternatives of $c \neq 3$ have yet not been ruled out a shift in r_∞ is still plausible.

3.2.2 Prediction of the expected error and its reduction

The question now is how well those predictions together are estimating r_t and r_{t+1} (and therefore the reduction in expected error which in turn determines the stopping time). Evaluating the latter is important since the prediction of the current error given ξ_t may be reasonably good, while the prediction for $t + 1$ given ξ_t may not be.

Fig. 3.9 (left column) shows the weighted average in absolute difference between the prediction of the expected error at time t and its true value.

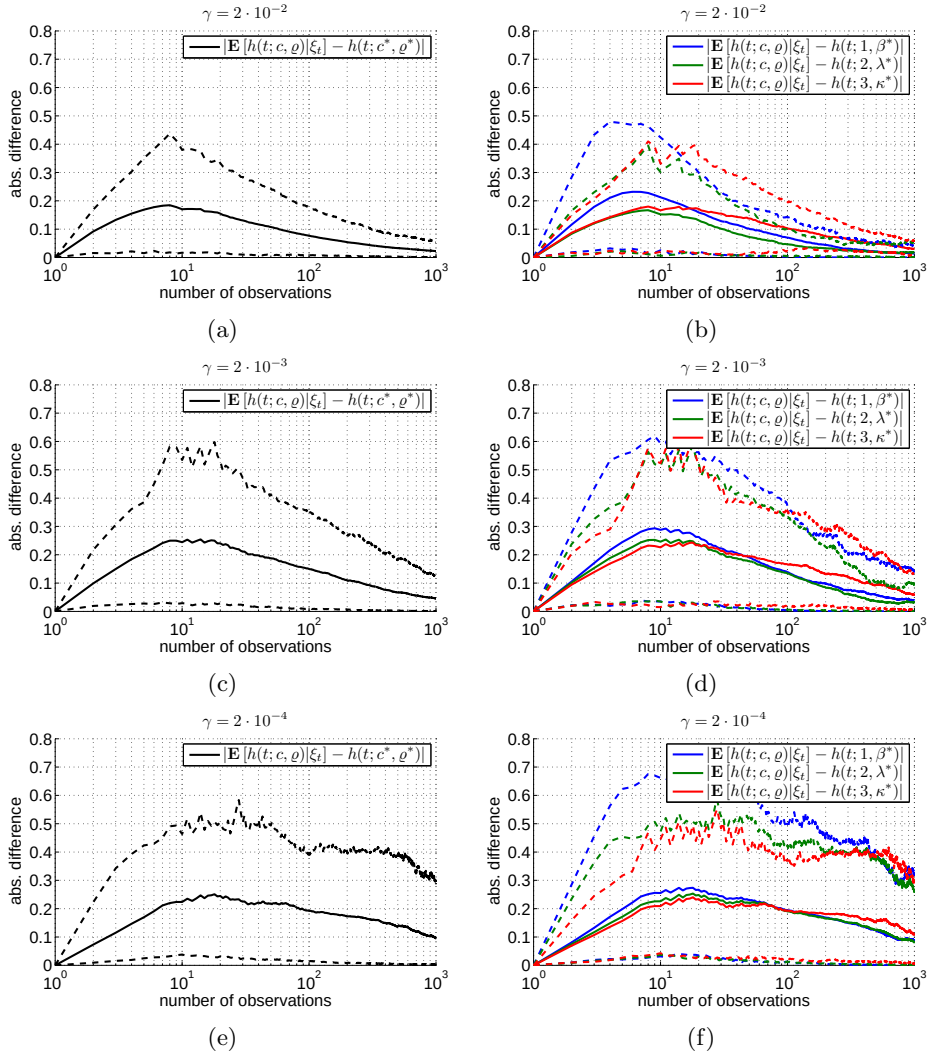


Figure 3.7: Results as obtained from 10 runs over all $\omega^* \in \Omega^*_\gamma$. The left column shows the weighted average in absolute difference between the predicted convergence and the target value as a function of t given $\gamma \in \{2 \cdot 10^{-2}, 2 \cdot 10^{-3}, 2 \cdot 10^{-4}\}$. The right column shows the corresponding unweighted averages given the different Ω^*_{γ,c^*} with $c^* = 1, 2, 3$. The dashed lines denote the $5th$ and $95th$-percentiles.

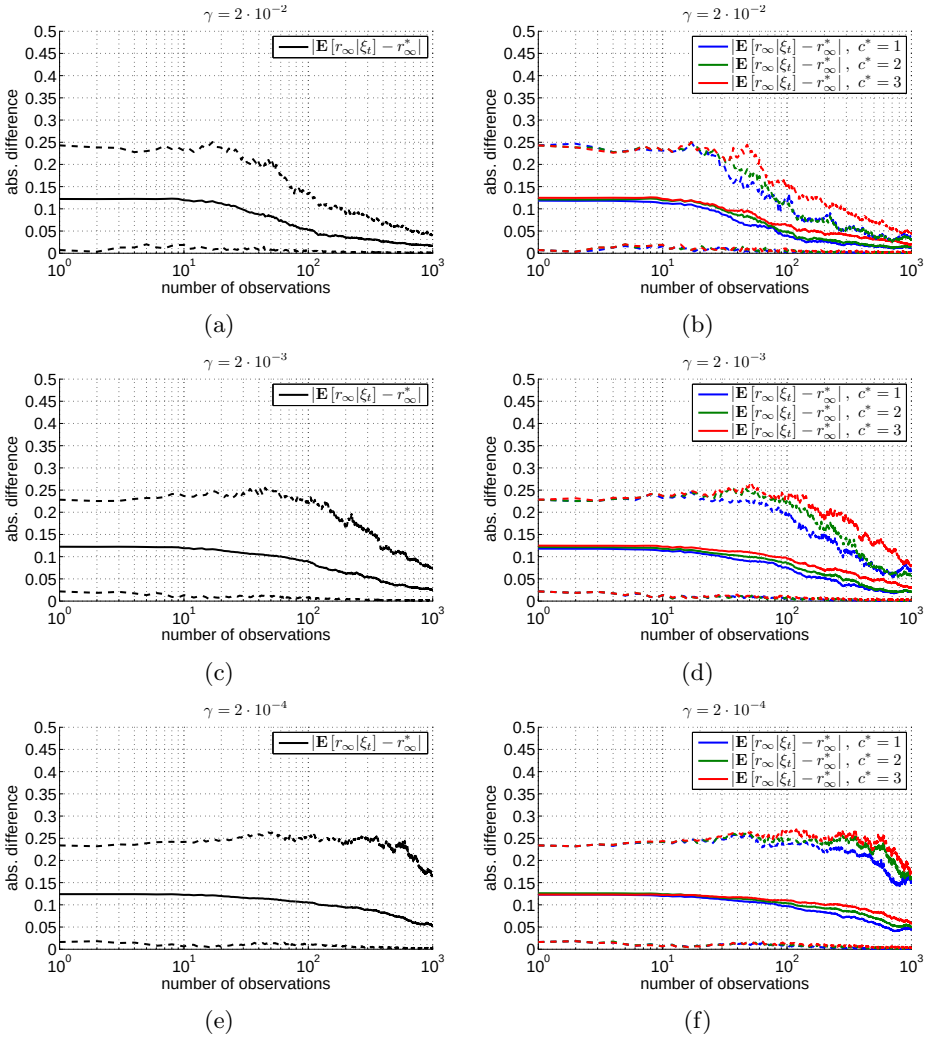


Figure 3.8: Results as obtained from 10 runs over all $\omega^* \in \Omega^*_\gamma$. The left column shows the weighted average in absolute difference between the predicted final error and the target value as a function of t given $\gamma \in \{2 \cdot 10^{-2}, 2 \cdot 10^{-3}, 2 \cdot 10^{-4}\}$. The right column shows the corresponding unweighted averages given the different Ω^*_{γ,c^*} with $c^* = 1, 2, 3$. The dashed lines denote the $5th$ and $95th$-percentiles.

Corresponding to our findings above, the average goes to zero as t increases while progress becomes faster as γ increases. Investigating these results in more detail given different c^* (Fig. 3.9, right column) we find that the predictions of the expected error are almost equally good after around 20 to 50 observations. For smaller values of t the predictions are worst given $c^* = 1$ since such targets converge fastest. It is particularly interesting perhaps to note that the predictions for $t + 1$ given ξ_t are very similar as shown by Fig. 3.10. Thus our model seems to produce reasonable estimates of the error after a sensible amount of observations.

We investigate this claim more formally by evaluating the resulting weighted average in absolute difference between the predicted reduction in expected error and its true value (Fig. 3.11). As before the overall progress (left column) becomes faster as γ increases. Although the $95th$-percentiles shown on the right are worst given $c^* = 1$, they are however still in the same order of magnitude as those given $c^* = 2, 3$. Similarly the performance at large values of t given $c^* = 3$ is also within the same order of magnitude as for $c^* = 1, 2$. Thus the reduction in expected error becomes reasonably small and we can hope to obtain sensible stopping times, even for small values of γ.

3.2.3 Expected stopping time and expected costs

Finally we evaluate the performance of OBSV by its expected stopping times and expected costs, and compare them to their optimal values. More specifically, we compare $t_\gamma = \mathbf{E}[T_\gamma \mid F, Q_F(\gamma), \langle \mathbf{z}_t \rangle]$ to the stopping time t^*_γ of the **oracle**. The latter is defined simply as the stopping time that minimises v_e when q_t is known. We will use v_e and v^*_e as a shorthand for the expected costs of OBSV and of the oracle.

For reference we plot the average in t_γ and t^*_γ (Fig. 3.12(a)) and note that OBSV never takes more than 3 observations when $\gamma \geq 3 \cdot 10^{-2}$ due to the prior. Furthermore OBSV follows the stopping times of the oracle on average within the same order of magnitude while the $5th$- and $95th$-percentile are well within the oracle's. However as is indicated by the $5th$-percentiles for $\gamma \to 0$, OBSV is late by more than an order of magnitude upon early stopping targets which is also due to its prior. The corresponding averages in absolute difference between t_γ and t^*_γ are shown in Fig. 3.12(c) - 3.12(d). As can be seen by the $95th$-percentiles, t_γ deviates by up to around 400 from t^*_γ when $c^* = 1$. Investigating the posterior beliefs we found that this is due to the slow exponential targets, where faster alternatives of $c > 1$ together with an overestimate of r^*_∞ indicate convergence and therefore lead to an early stopping of OBSV. However those maximum values are reached only for small γ where the contribution of the stopping time itself to the cost is comparably small unless $r_{t^*_\gamma} \to 0$.

Finally we evaluate the average difference between the expected costs of

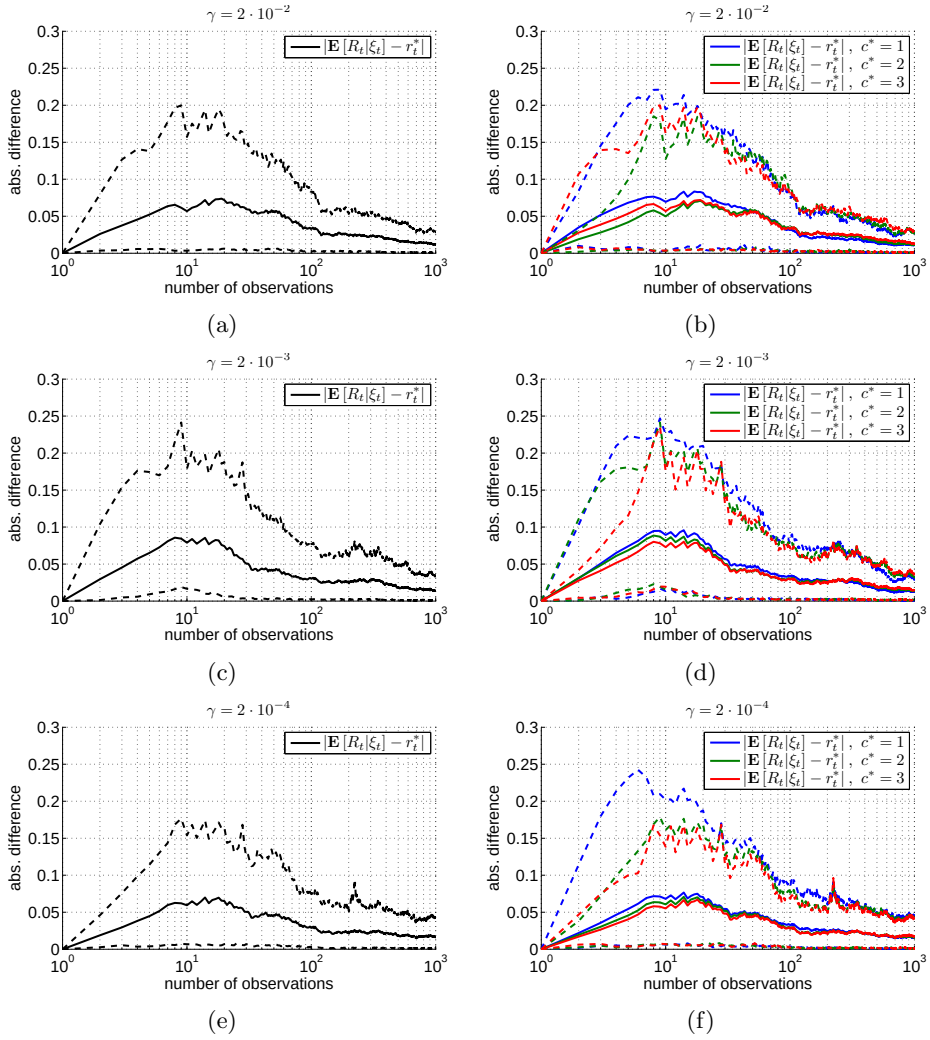


Figure 3.9: Results as obtained from 10 runs over all $\omega^* \in \Omega^*_\gamma$. The left column shows the weighted average in absolute difference between the predicted error and the target value as a function of t given $\gamma \in \left\{2 \cdot 10^{-2}, 2 \cdot 10^{-3}, 2 \cdot 10^{-4}\right\}$. The right column shows the corresponding unweighted averages given the different Ω^*_{γ,c^*} with $c^* = 1, 2, 3$. The dashed lines denote the $5th$ and $95th$-percentiles.

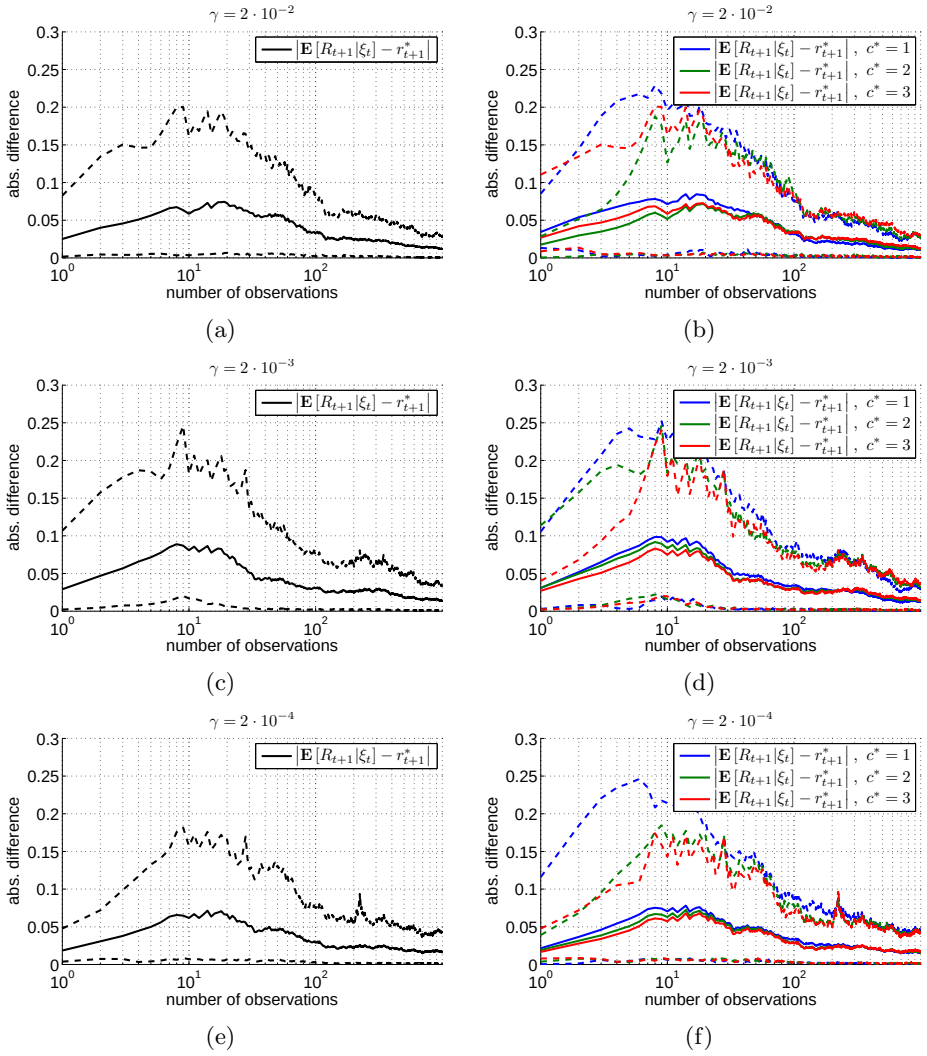


Figure 3.10: Results as obtained from 10 runs over all $\omega^* \in \Omega^*_\gamma$. The left column shows the weighted average in absolute difference between the predicted error for $t+1$ given ξ_t and the target value as a function of t given $\gamma \in \left\{2 \cdot 10^{-2}, 2 \cdot 10^{-3}, 2 \cdot 10^{-4}\right\}$. The right column shows the corresponding unweighted averages given the different Ω^*_{γ,c^*} with $c^* = 1, 2, 3$. The dashed lines denote the $5th$ and $95th$-percentiles.

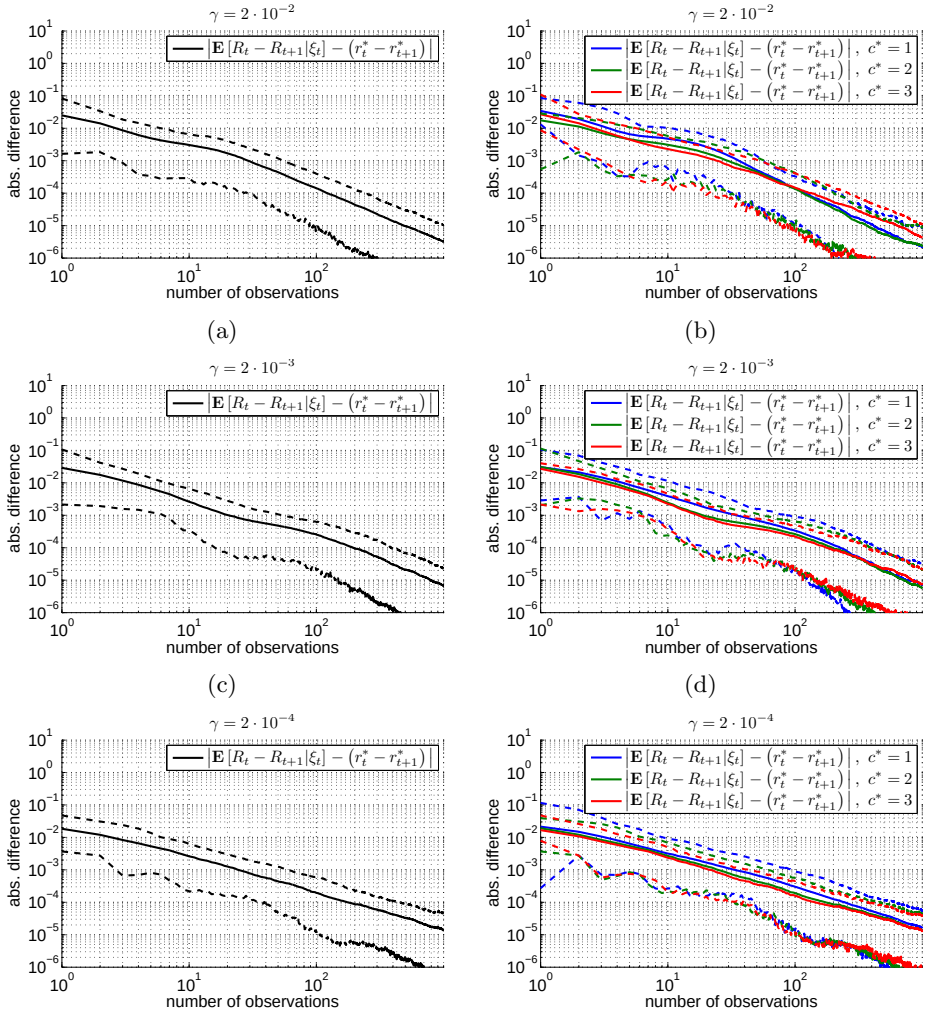


Figure 3.11: Results as obtained from 10 runs over all $\omega^* \in \Omega_\gamma^*$. The left column shows the weighted average in absolute difference between the predicted reduction in error given ξ_t and the target value as a function of t given $\gamma \in \left\{2 \cdot 10^{-2}, 2 \cdot 10^{-3}, 2 \cdot 10^{-4}\right\}$. The right column shows the corresponding unweighted averages given the different Ω_{γ,c^*}^* with $c^* = 1, 2, 3$. The dashed lines denote the $5th$ and $95th$-percentiles.

OBSV and the oracle where we scale each cost difference by the costs of the oracle before averaging. Thus we evaluate the ratio $(v_e - v_e^*)/v_e^* = v_e/v_e^* - 1$. As can be seen by Fig. 3.12(e) the $95th$-percentiles of the additional costs induced by using OBSV instead of the oracle averaged over all $\omega \in \Omega_\gamma^*$ is bounded from above by around 10% for values of $\gamma \leq 3 \cdot 10^{-2}$ where OBSV takes more than 3 observations. For more reasonable values of $\gamma \leq 2 \cdot 10^{-3}$ where the oracle takes at least 20 observations on average, the average in additional costs is around 3%. For larger values of γ however the advantage of the oracle becomes larger due to its full information.

We investigate these results in more detail given different c^*. As shown by Fig. 3.12(f) for $c^* = 2, 3$ the additional costs of OBSV are around 2% to 3% on average while for $c^* = 1$ the average costs are around 8%. We found that the results around the $95th$-percentile for $c^* = 1$ correspond to *fast converging targets of low final error*. For such targets even a small deviation from the optimal stopping time leads to a comparbly larger ratio in cost as the same deviation from the optimal stopping would do given other targets. Thus, corresponding to our findings about stopping times reported above, the comparably larger ratio in cost for $c^* = 1$ is due to (a) late stopping of OBSV upon fast converging targets of small final error (see $5th$-percentiles in Fig. 3.12(a)) and (b) early stopping of OBSV upon slow exponential targets (see $95th$-percentiles in Fig. 3.11(d)).

It is worthwhile to mention that we performed the same experiments using $m_t = 3, 10$ observations per iteration and obtained almost identical results. That is, when m_t increases the predictions of our model as well as the stopping time and cost of OBSV improve *when plotted against the number of iterations* t. However since we take more observations per iteration the results basically remain the same *when plotted against the number of observations* $m_t \cdot t$. The only real difference is that for increasing m_t the immediate stop of OBSV mentioned above already occurs for smaller values of γ. However for $\gamma \to 0$ the ratios in cost become almost the same as for $m_t = 1$.

3.2.4 Maximum a posteriori predictions

We compare the results of OBSV using Bayesian predictions to those for maximum a posteriori (MAP) predictions which we will denote by

$$\hat{\omega} = \underset{\omega \in \Omega_\gamma}{\arg\max} \; \mathbf{P}(\omega \mid \xi_t)$$

and imply the dependence on ξ_t on the right side unless stated otherwise. We will denote the MAP prediction of the convergence by $h(t; \hat{c}, \hat{\varrho})$, OBSV itself using MAP predictions by $\widehat{\text{OBSV}}$ and its stopping time and costs by $\hat{t}_\gamma$ and $\hat{v}_e$ respectively.

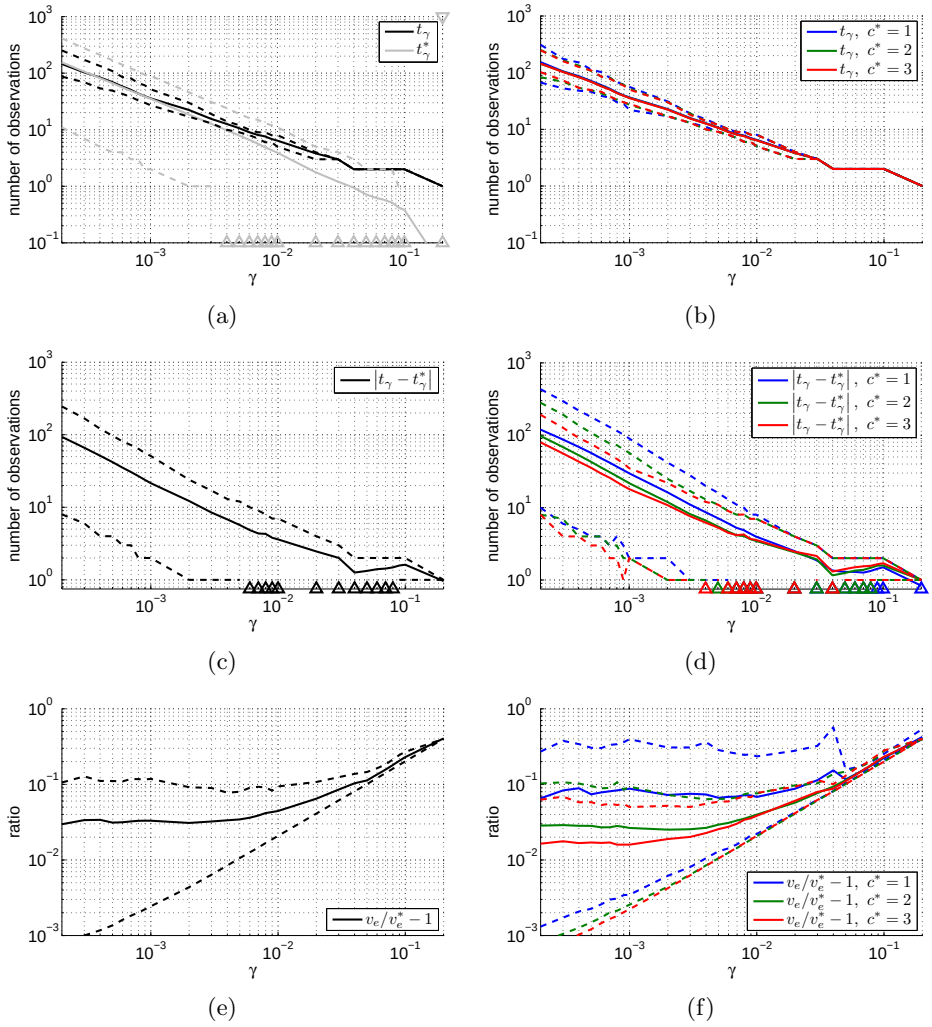


Figure 3.12: Results as obtained from 10 runs over all $\omega^* \in \Omega^*$. Fig. 3.12(a) shows the weighted average number of observations taken by OBSV and by the oracle as a function of γ. Fig. 3.12(b) shows the average number of observations taken by OBSV given different c^*. Fig. 3.12(c) - 3.12(d) show the (weighted) absolute differences between the number of observations taken by OBSV and by the oracle averaged over $\omega \in \Omega^*_\gamma$ and over $\omega \in \Omega^*_{\gamma,c}$. Fig. 3.12(e) - 3.12(f) shows the (weighted) average of the ratio in cost. The dashed lines denote the $5th$ and $95th$-percentiles where a triangle denotes a value of zero.

The MAP predictor is a common baseline for learning algorithms [48]. In our case it also allows us to confirm uncertainty within our model. That is, if the models of the expected learning curves would (a) overlap weakly over all $\Omega_{\gamma,c},\ c = 1, 2, 3$ or (b) would be far apart from each other within each $\Omega_{\gamma,c}$, then the identification of the target would be easy. Thus the experimental setup would lack of uncertainty and we would expect that the performance of the MAP predictions would be almost as good as the Bayesian ones.

It is important to note that, given our model, the MAP prediction corresponds to the expected error of *the most probable learning curve* $g\,(t; \hat{\omega})$ *given* ξ_t whereas the full Bayesian prediction corresponds to *the most probable expected error given* ξ_t, i.e. a linear combination of all $\omega \in \Omega_\gamma$. Therefore, even if the observed learning curve given some real F and $\mathcal{D}$ may not be in Ω_γ we can still hope to predict it reasonably well. However, in our experimental setup we consider only such artificial target beliefs ξ^* which correspond to a specific target model $\omega^* \in \Omega^*$ and therefore give the MAP predictor a fair chance.

As can be seen by Fig. 3.13, the $95th$-percentiles of both, (1) the weighted average in absolute difference between the predicted MAP convergence and its true value and (2) that between the predicted MAP final error and its true value are comparably larger than those given the full Bayesian setting. This discrepancy becomes largest after around 10 to 20 observations. Afterwards the MAP predictions approach the Bayesian ones but nevertheless remain worse. This has the following reason: investigating the posteriors we found that the MAP-predictor is overfitting the data when only few observations are given.

Next we evaluate the impact of this finding on the prediction of the current error and on its predicted reduction. As can be seen by the averages and $95th$-percentiles shown in Fig. 3.14 the performance of $\widehat{\text{OBSV}}$ remains inferior. Furthermore we note that the MAP predictions can be superior compared to the Bayesian ones as shown by the $5th$-percentiles of the estimated reduction in error after around 20 observation. Such superiority for small values of t is related to the event of incidentally overfitting the data using an appropriate model or even the target itself (for example see the $5th$-percentiles for small t in Fig. 3.14(f)).

Furthermore for small values of t we frequently observed that as long as $z_t = 0$ the MAP-predictor selects rather fast converging models, whereas it directly switches to models of slow convergence and large final error upon $z_t = 1$. For many targets we observe that if t is small, r_t is of the same order of magnitude as r_0. Therefore it is likely that $z_t = 1$ at least once for such small values of t. In consequence $\widehat{\text{OBSV}}$ regularly stops too early as indicated by Fig. 3.15(a). As shown there the $95th$-percentiles of the number of observations taken by $\widehat{\text{OBSV}}$ are regularly smaller than the weighted average of the number of observations taken by the oracle. Therefore the

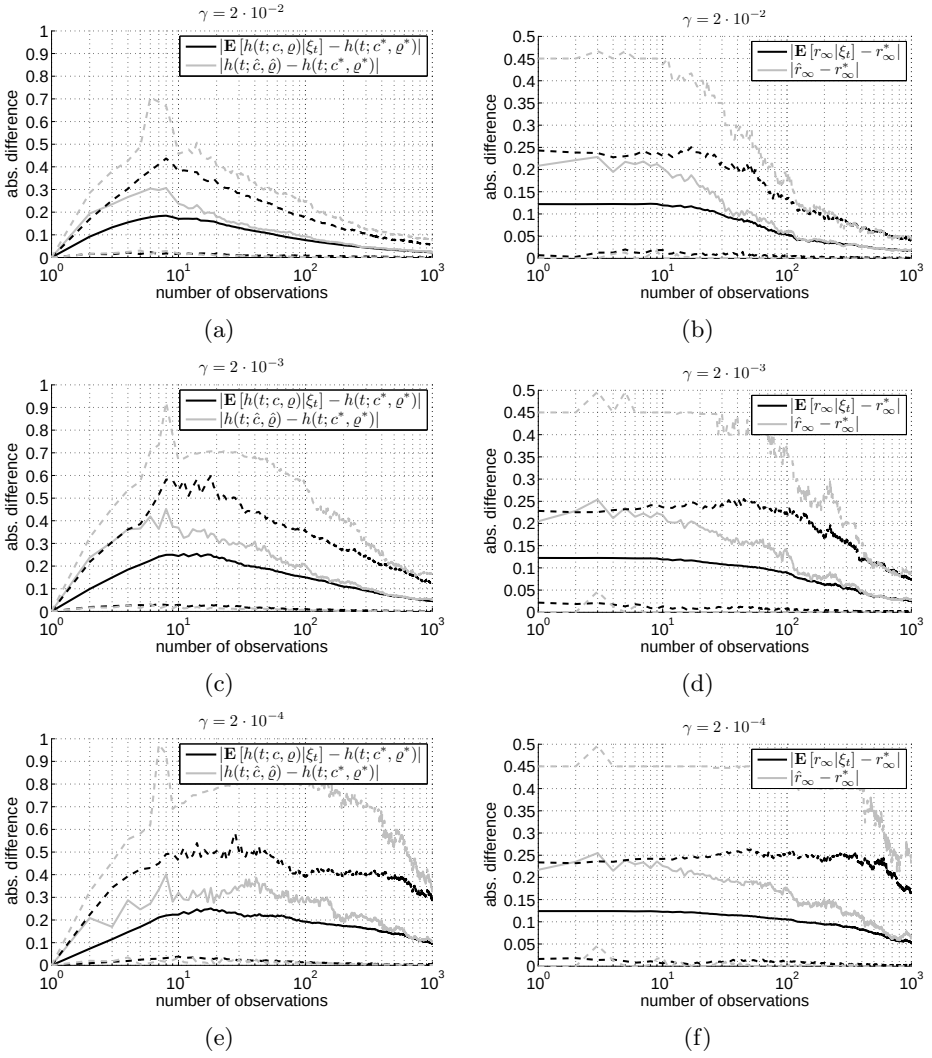


Figure 3.13: Results as obtained from 10 runs over all $\omega^* \in \Omega^*_\gamma$. The left column shows the weighted average in absolute difference between the predicted convergence and the target value as a function of t given $\gamma \in \left\{2 \cdot 10^{-2}, 2 \cdot 10^{-3}, 2 \cdot 10^{-4}\right\}$. The right column shows the weighted average in absolute difference between the predicted final error and the target value. The dashed lines denote the $5th$ and $95th$-percentiles.

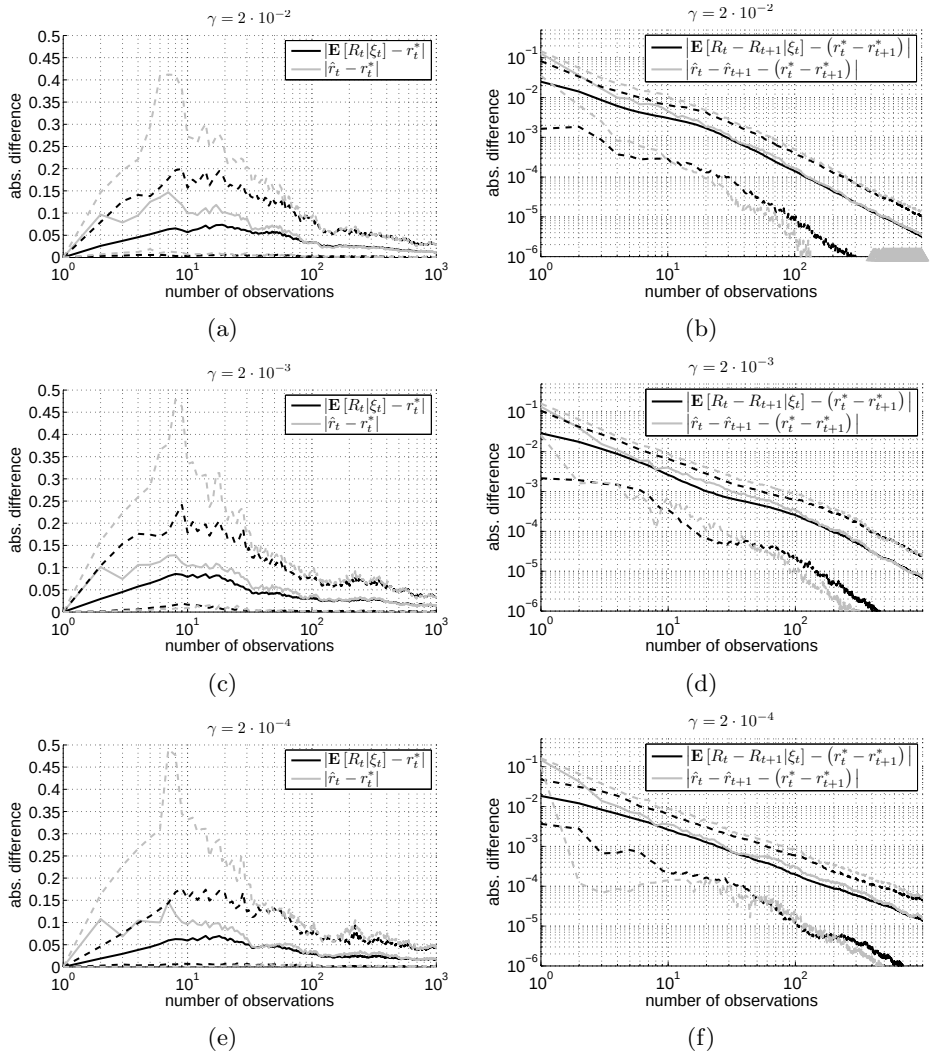


Figure 3.14: Results as obtained from 10 runs over all $\omega^* \in \Omega_\gamma^*$. The left column shows the weighted average in absolute difference between the predicted error at time t and the target value as a function of t given $\gamma \in \{2 \cdot 10^{-2}, 2 \cdot 10^{-3}, 2 \cdot 10^{-4}\}$. The right column shows the weighted average in absolute difference between the predicted reduction in error and the target value. The dashed lines denote the $5th$ and $95th$-percentiles where a triangle denotes a value of zero.

weighted average in absolute difference of number of observations taken between $\widehat{\text{OBSV}}$ and the oracle is frequently larger than that between OBSV and the oracle (Fig. 3.15(b)). Hence the additional costs by using $\widehat{\text{OBSV}}$ instead of the oracle are on average larger by up to an order of magnitude than those of using OBSV instead of the oracle (Fig. 3.15(c)).

In conclusion these results (a) show that there is uncertainty about the models $\omega \in \Omega_\gamma$ and (b) confirm the commonly known superiority of Bayesian predictions over MAP predictions under uncertainty.

3.2.5 A naive stopping algorithm

Finally we compare OBSV to a *naive stopping algorithm*. The latter will ignore all observations of the error and, since $r_t - r_{t+1} \leq r_0 - r_\infty \leq r_0$,

$$\text{stop when } t > \tfrac{r_0}{\gamma}.^2$$

We will use $\bar{t}_\gamma$ and $\bar{v}_e$ as shorthands for its stopping time and the expected cost.

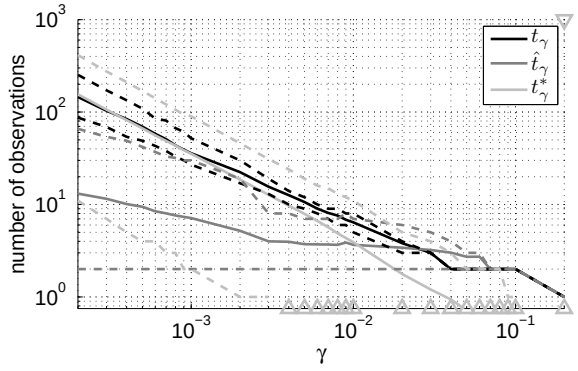
Fig. 3.16(a) shows the the average number of observations taken by the naive stopping algorithm, OBSV and the oracle. There we can see that the naive algorithm always stops later than the oracle or OBSV. Therefore we omit the figures for the absolute differences in stopping times. Investigating the additional costs (Fig. 3.16(b)) we find that the performance of OBSV is frequently better by more than an order of magnitude when compared to that of the naive stopping criterion.

3.3 UCI data sets

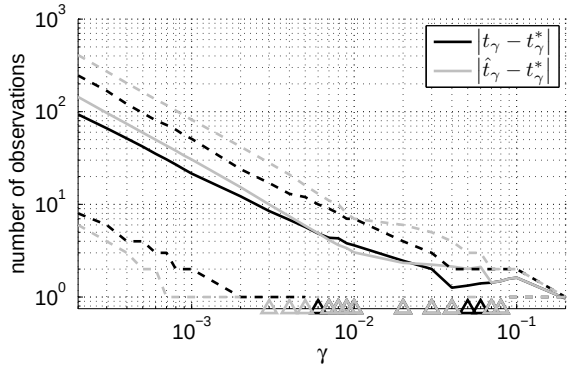
The main purpose of this section is to evaluate the performance of the OBSV stopping algorithm upon real data sets and learning algorithms. First we relate the performance of OBSV to the baselines introduced in Sec. 3.2.3 - 3.2.5 given different sampling strategies (Sec. 3.3.1). Then we compare these sampling strategies to each other (Sec. 3.3.2). Finally we compare different combinations of sampling strategies and learning algorithms (Sec. 3.3.3).

Following [8], we plot performance curves for a range of values of γ, utilising multiple runs of cross-validation in order to assess the sensitivity of the results to the data. Given some specific randomisation of the data and some specific split of the corresponding folds into a training set $\mathcal{D}$ and a test set $\mathcal{D}_E$, we split $\mathcal{D}$ itself into random and active sampling sets whenever appropriate. Such an experiment will be referred to as a *run* unless stated otherwise.

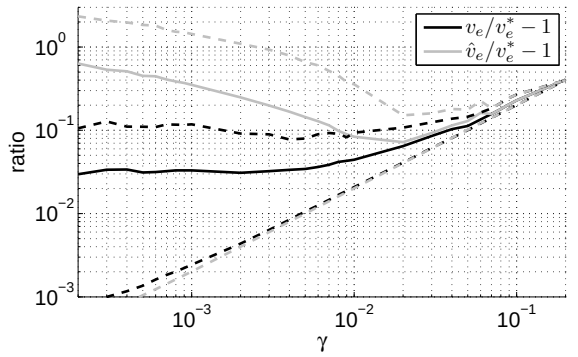
[2] More precisely when $|\mathcal{D}_T| > r_0/\gamma = 0.5/\gamma$ it updates the classifier (Alg. 5, Step 13) and stops immediately without taking any observations of its error (Alg. 5, Step 6).



(a)

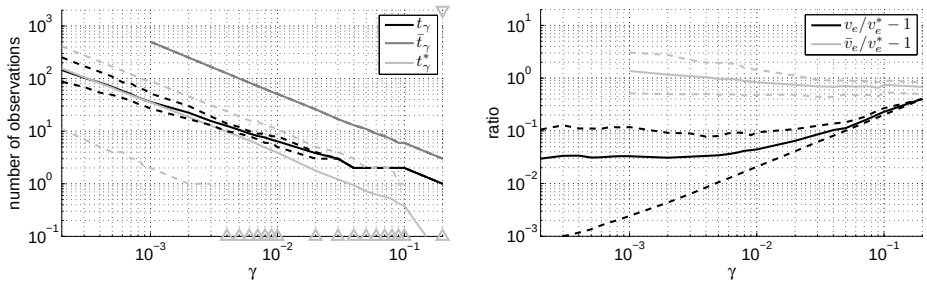


(b)



(c)

Figure 3.15: Results as obtained from 10 runs over all $\omega^* \in \Omega^*$. Fig. 3.15(a) shows the weighted average in number of observations taken by OBSV, by $\widehat{\text{OBSV}}$ and by the oracle as a function of γ. Fig. 3.15(b) shows the weighted average in absolute difference of observations taken between OBSV and the oracle and between $\widehat{\text{OBSV}}$ and the oracle over $\omega \in \Omega^*_\gamma$. Fig. 3.15(c) shows the weighted average of the ratio in cost. The dashed lines denote the $5th$ and $95th$-percentiles where a triangle denotes a value of zero.



(a) (b)

Figure 3.16: Results as obtained from 10 runs over all $\omega^* \in \Omega^*$. Fig. 3.16(a) shows the weighted average in number of observations taken by OBSV, by the naive stopping criterion and by the oracle as a function of γ. Fig. 3.16(b) shows the weighted average of the ratio in cost between the naive stopping criterion and the oracle, and between OBSV and the oracle. The dashed lines denote the $5th$- and $95th$-percentiles where a triangle denotes a value of zero.

As we generally do not have ξ^* given real data, we cannot compare the predictions of our model to a target convergence $h\,(t; c^*, \varrho^*)$ or target final error r^*_∞ and therefore directly evaluate the stopping times and costs. For every value of γ we obtain a different stopping time t_γ from each run. We then calculate $v_e(\gamma, t_\gamma, F)$ as given in Eq. 2.2 on the corresponding test set of the run. Thus we define the **oracle** stopping time as the stopping time minimising the cost as measured on the independent test set for a specific run. By examining the averages and extreme values over all runs we are able to estimate the sensitivity of the results to the data.

For *random* sampling, we simply query unlabelled examples without replacement from $\mathcal{D}_R$. For the *mixed sampling* procedure, we actively query an additional label for the example from $\mathcal{D}_A$ closest to the decision boundary of the current classifier, also without replacement. This strategy relies on the assumption that those labels are most informative [1], [43], [38] and therefore convergence will be faster. For *active sampling* we only actively query examples from $\mathcal{D}_A$ in Step 6 instead of Step 12 and use those to determine whether to stop or not. Although this clearly violates the i.i.d assumption for the validation data, we are interested in OBSV's behaviour under such circumstances. Stopping times and differences in cost are shown for a set of γ values. Note that our comparison of any two stopping algorithms $Q_F(\gamma), Q_F(\gamma)\prime$ covers only such values of γ where $\max\,(t_\gamma, t_{\gamma\prime}) < m$, since $m = |\mathcal{D}| < \infty$. Furthermore we plot the average test error curves for reference.

For the experiments we use two data sets from the UCI repository [5]:

the Wisconsin breast cancer data set (`wdbc`) with 569 examples and the spambase database (`spam`) with 4601 examples. We evaluate `wdbc` and `spam` using 20 randomised runs of 3-fold stratified cross-validation. Hence we obtain a total of 60 runs for each dataset. The classifiers used are AdaBoost with 100 decision stumps as base hypotheses and the classical Perceptron learning algorithm using 20 iterations over the training data (see Sec. A.1, Alg. 6). For the Perceptron the data sets were normalised by PCA.

3.3.1 Comparing OBSV to the baselines given different sampling strategies

First we compare the performance of OBSV given the different sampling strategies. This is done by examining the costs and stopping times when compared to (a) $\widehat{\text{OBSV}}$, (b) the the naive stopping algorithm and (c) the the oracle.

Random sampling

We evaluate the performance of OBSV for random sampling on both data sets. As a reference we plot the realised error q_t as measured on the test set[3]. Fig. 3.17(a) - 3.17(b) shows the average in test error over all runs.

As can be seen by Fig. 3.17(c) - 3.17(d), $\widehat{\text{OBSV}}$ regularly stops too early, whereas for OBSV the $5th$- and $95th$-percentiles of the stopping times are well within the oracle's. As already observed for artificial data, the reason for the early stopping of $\widehat{\text{OBSV}}$ is overfitting. The greater variation of the oracle on `wdbc` is partially a result of the small dataset. Since we can only measure an error in quanta of $1/|\mathcal{D}_E|$, any actual performance gain lower than this will be unobservable. More specifically, the oracle exploits local minima of q_t which become more prominent as the size of the test set decreases. We expect the oracle's behaviour would be smoother with larger data sets. This can actually be observed for `spam`.

In consequence the additional costs of using OBSV instead of the oracle are always lower than the those for $\widehat{\text{OBSV}}$ or the naive stopping algorithm by up to an order of magnitude (Fig. 3.17(e) - 3.17(f)). As can be seen on `spam` only for $\gamma \leq 2 \cdot 10^{-5}$, when around 100 observations have been taken, the MAP predictions stabilise and their additional costs approach those of the Bayesian ones. Furthermore, as $\bar{t}_\gamma$ is far away from t^*_γ, both OBSV and $\widehat{\text{OBSV}}$ yield comparably better stopping times and therefore ratios in cost.

As can be seen by Fig. 3.17(a) - 3.17(b) the true final error r^*_∞ is low and therefore, compared to larger values of r^*_∞, a deviation from the oracle's stopping time yields a larger ratio in cost. Thus the results for `spam` are comparable to those for artificial observations (Sec. 3.2). For `wdbc` however

[3] also known as the *test error*

the cost ratios shown are larger due to the advantage of the oracle upon smaller test sets as discussed above.

Mixed sampling.

Next we compare OBSV to the baselines given mixed sampling. The corresponding test errors are shown in Fig. 3.18(a) - 3.18(b). As before the stopping times of OBSV are well within the oracle's which indicates a satisfactory performance. $\widehat{\text{OBSV}}$ however regularly stops too early due to overfitting. Similar to random sampling, the additional costs of using OBSV instead of the oracle are always lower than those of $\widehat{\text{OBSV}}$ or the naive stopping algorithm by up to around an order of magnitude (Fig. 3.18(e) - 3.18(f)).

Active sampling.

Finally we compare OBSV to the baselines given active sampling. The corresponding test errors are shown in Fig. 3.19(a) - 3.19(b). As can be seen by Fig. 3.19(c) - 3.19(d) the stopping times of OBSV are not as smooth as for the other sampling strategies. More specifically, the changes of t_γ can be divided into 3 phases depending on γ:

1. If $\gamma \to \left(7 \cdot 10^{-4}\right)^+$ for `wdbc` or $\gamma \to \left(10^{-4}\right)^+$ for `spam` OBSV stops a bit earlier than the oracle on average. That is, OBSV takes up to around 50 observations from `wdbc` and up to around 200 observations from `spam`. According to the strategy of sampling along at the current decision boundary, we found that the error v_t observed on those examples is frequently much higher than q_t, i.e. than the error given i.i.d. examples. Thus OBSV assigns more weight to slow convergence models of large final error. In turn the predicted reduction in error is too low and therefore OBSV stops too early.

2. With $3 \cdot 10^{-4} \leq \gamma < 7 \cdot 10^{-4}$ for `wdbc` or $5 \cdot 10^{-5} \leq \gamma < 10^{-4}$ for `spam` the slope of the stopping times increases and on average OBSV stops later than the oracle. By investigating the margins we find that the informative examples are exhausted for $t \approx t_\gamma$. Thus the error observed at such t corresponds more to the realised error q_t than before. In consequence the predicted reduction in error increases and OBSV takes more observations than necessary. Therefore the additional costs shown in Fig. 3.19(e) - 3.19(f) actually increase suddenly for these values of γ.

3. For $\gamma < 3 \cdot 10^{-4}$ given `wdbc` and $\gamma < 5 \cdot 10^{-5}$ given `spam` the slope of the stopping times becomes smaller again. The reason is that now, as even more observations have been taken, OBSV has adapted to

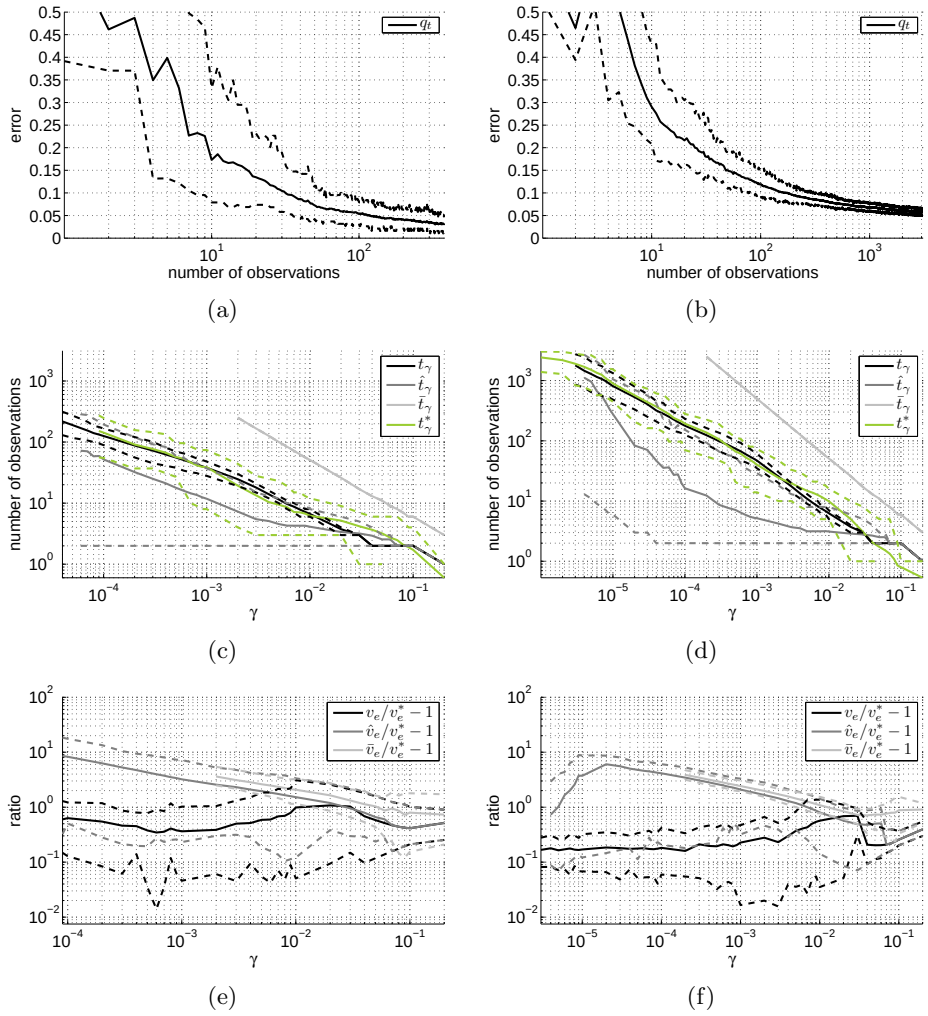


Figure 3.17: Results for **random sampling** on the `wdbc` (left column) and the `spam` data (right column) as obtained from 60 runs of AdaBoost using 100 decision stumps as weak classifiers. The first row (Fig. 3.17(a) - 3.17(b)) shows the realised error as measured on the test set averaged over of all runs. The second row (Fig. 3.17(c) - 3.17(d)) shows the average of the expected number of observations taken by OBSV, $\widehat{\text{OBSV}}$, the naive stopping algorithm and the oracle. The third row (Fig. 3.17(e) - 3.17(f)) shows the average of the ratio in cost w.r.t. the oracle. The dashed lines denote the $5th$- and $95th$-percentiles.

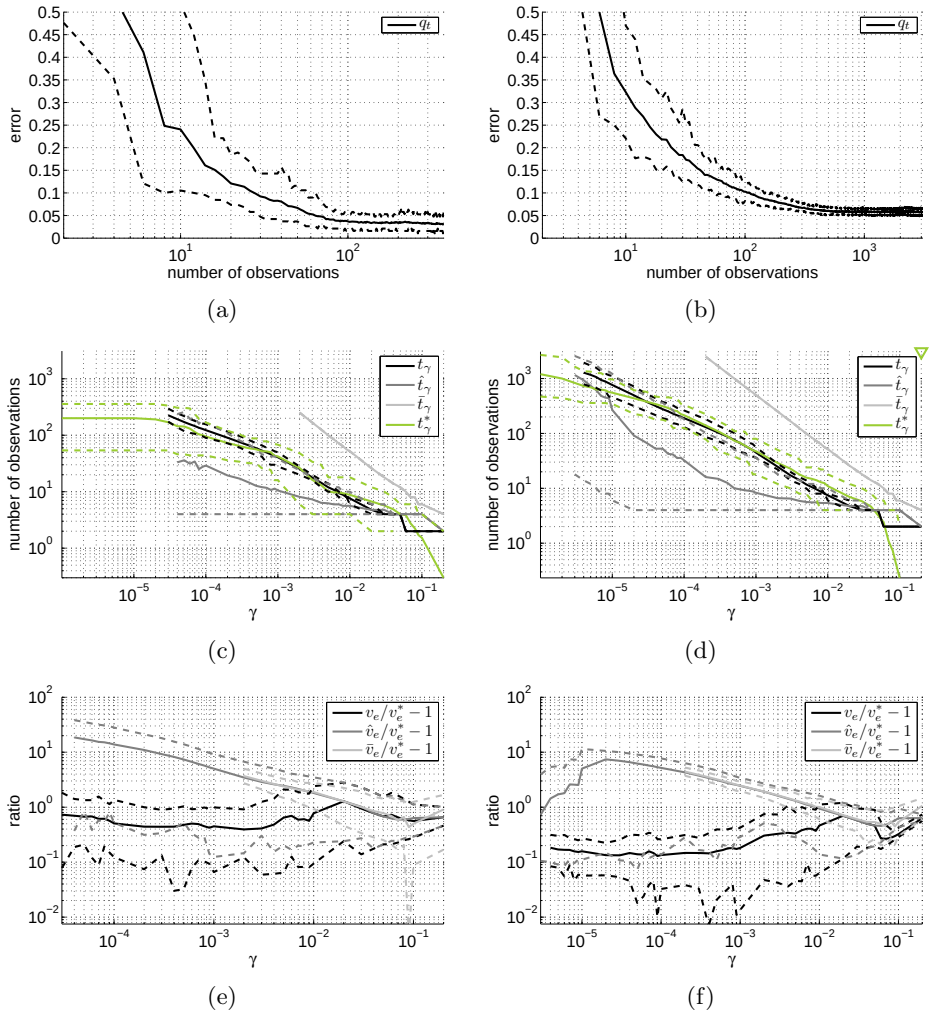


Figure 3.18: Results for **mixed sampling** on the `wdbc` (left column) and the `spam` data (right column) as obtained from 60 runs of AdaBoost using 100 decision stumps as weak classifiers. The first row (Fig. 3.18(a) - 3.18(b)) shows the realised error as measured on the test set averaged over of all runs. The second row (Fig. 3.18(c) - 3.18(d)) shows the average of the expected number of observations taken by OBSV, $\widehat{\text{OBSV}}$, the naive stopping algorithm and the oracle. The third row (Fig. 3.18(e) - 3.18(f)) shows the average of the ratio in cost w.r.t. the oracle. The dashed lines denote the $5th$- and $95th$-percentiles.

the convergence of q_t towards r_∞. Therefore it predicts a comparably lower reduction in error than before.

We conclude that the violation of the i.i.d. assumption deteriorates the stopping times of OBSV in comprehensible manner. Nevertheless its performance remains superior compared to $\widehat{\text{OBSV}}$ and the naive stopping algorithm.

3.3.2 Comparing different sampling strategies

Finally we compare the three sampling strategies directly. As a reference the test errors are shown together in Fig. 3.20. We note that q_t decreases fastest for active sampling and that on average it is closely followed by the mixed strategy. Compared to the random strategy the faster convergence of q_t given mixed or active sampling is however, except for around 500 observations on `spam`, not very significant as the percentiles overlap strongly.

Therefore the stopping times of OBSV are around the same as shown in Fig. 3.21(a) - 3.21(b). It is important to note that the stopping times of the oracle also overlap strongly (Fig. 3.21(c) - 3.21(d)). We believe that the difference in stopping time between the sampling strategies would increase if the active selection procedure were more effective. Particularly we expect OBSV to produce more diverse stopping curves, similar to those shown in Sec. 3.2 for artificial observations given different classes of target convergence. However given the similar learning curves observed here OBSV produces similar stopping times as desired. Thus the slightly lower cost of the mixed strategy compared to the random strategy, as shown in Fig. 3.21(e) - 3.21(f), is mainly due to the difference in test error. However as will we see next, OBSV produces more diverse stopping curves when the learning curves differ significantly.

3.3.3 Comparing different combinations of sampling strategies and learning algorithms

Finally we provide a comparison of different combinations of sampling strategies and learning algorithms by utilising OBSV. This is straight forward given our framework. As we have seen before, OBSV produces similar stopping times given similar learning curves. To confirm the adaption of stopping times by OBSV we choose a setting which yields more diverse learning curves. More specifically, we compare AdaBoost using mixed sampling to the Perceptron using the random sampling strategy.

As can be seen by Fig. 3.22(a) - 3.22(b), on average learning proceeds much more slowly given the Perceptron with the random strategy compared to AdaBoost using mixed sampling. In consequence the average stopping times for AdaBoost given values of $\gamma \leq 10^{-3}$ are comparably smaller for both data sets (Fig. 3.22(c) - 3.22(d)). It is important to note that, with

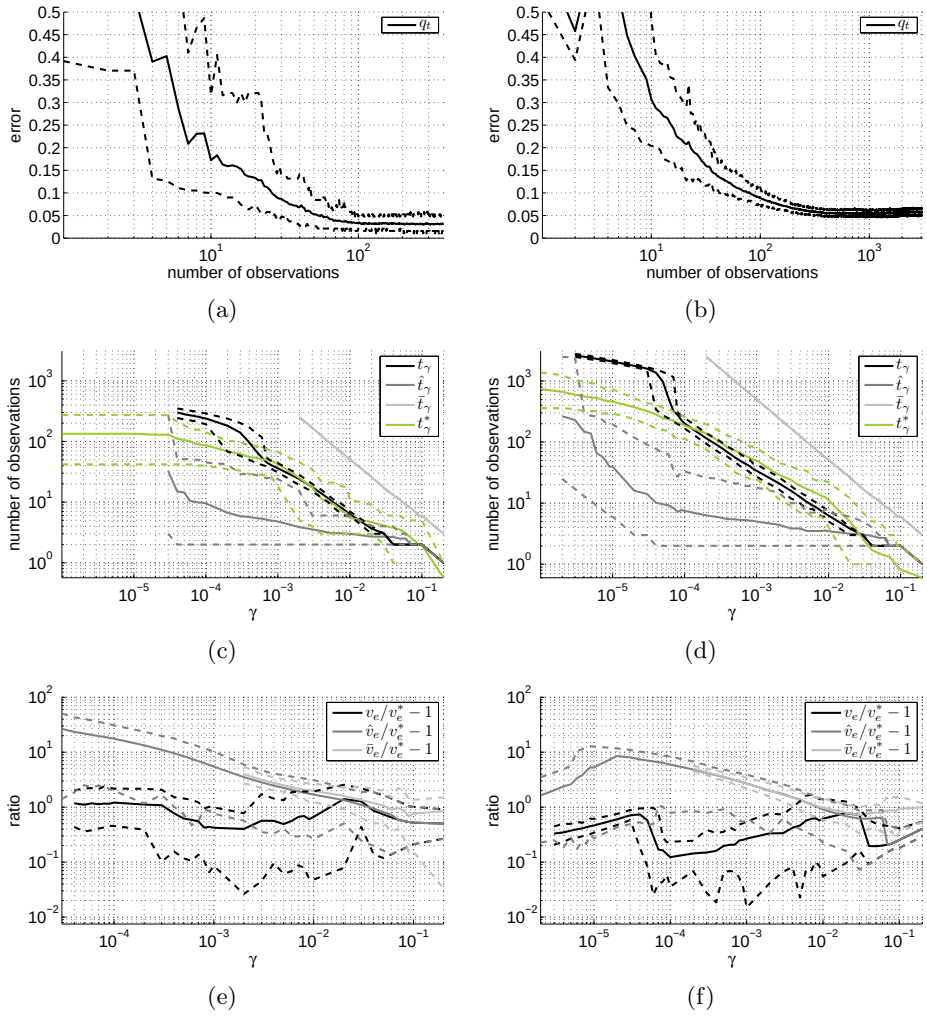


Figure 3.19: Results for **active sampling** on the `wdbc` (left column) and the `spam` data (right column) as obtained from 60 runs of AdaBoost using 100 decision stumps as weak classifiers. The first row (Fig. 3.19(a) - 3.19(b)) shows the realised error as measured on the test set averaged over of all runs. The second row (Fig. 3.19(c) - 3.19(d)) shows the average of the expected number of observations taken by OBSV, $\widehat{\text{OBSV}}$, the naive stopping algorithm and the oracle. The third row (Fig. 3.19(e) - 3.19(f)) shows the average of the ratio in cost w.r.t. the oracle. The dashed lines denote the $5th$- and $95th$-percentiles.

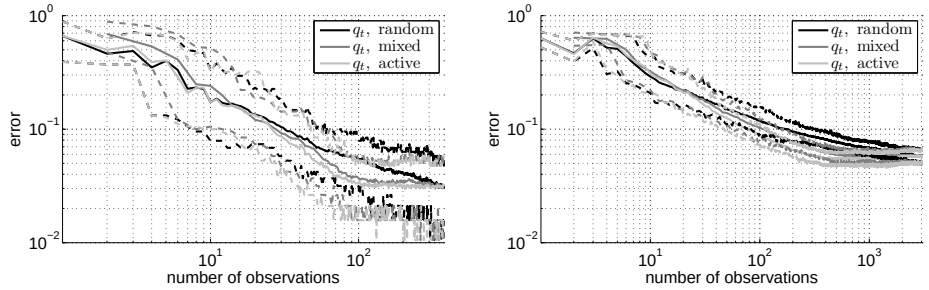


Figure 3.20: Results comparing **random, mixed and active sampling** on the `wdbc` (left column) and the `spam` data (right column) as obtained from 60 runs of **AdaBoost** using 100 decision stumps as weak classifiers. The figures show the average in test error over of all runs. The dashed lines denote the $5th$- and $95th$-percentiles.

reference to the stopping times of the oracle (Fig. 3.22(e) - 3.22(f)), this difference is justified. Therefore the averages in cost of AdaBoost utilising the mixed sampling strategy are also significantly smaller than the cost of the Perceptron using the random sampling procedure (Fig. 3.23(a) - 3.23(b)). Moreover these cost curves are very similar to those obtained by using the oracle (see Fig. 3.23(c) - 3.23(d) for comparison). Therefore we conclude that OBSV is reasonably adaptive and produces diverse stopping curves when the learning curves differ significantly.

3.4 PASCAL VOC Challenge 2006 data sets

The following section demonstrates the practical use of combining an image categorisation framework as described in Chap. 1 with OBSV proposed in Chap. 2. We will evaluate the performance given image data from `VOC06` according to two concrete scenarios. Thereby we will show (a) how to choose γ in practice, (b) that OBSV performs better than the baselines, and (c) that mixed and random sampling perform almost the same.

3.4.1 The choice of γ given concrete scenarios

In practice the perception of costs depends on the actual goal for using such a framework. One obvious goal could be to minimise the *monetary costs* of its implementation. In this case one could chose γ depending on the monetary costs M_L of labelling one example and the monetary costs M_R of the misclassification of all examples that we expect during the systems lifetime. That is, the total cost, as given by Eq. 2.1, could be used to measure

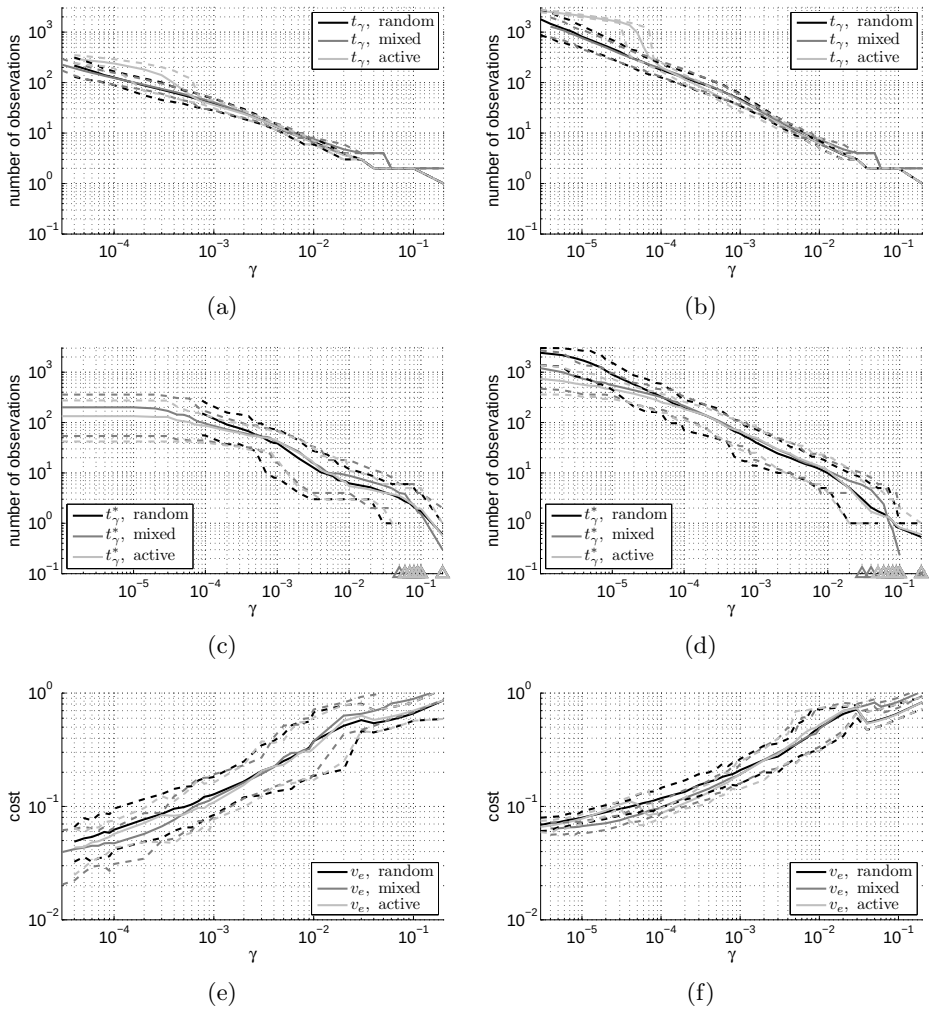


Figure 3.21: Results comparing **random, mixed and active sampling** on the `wdbc` (left column) and the `spam` data (right column) as obtained from 60 runs of AdaBoost using 100 decision stumps as weak classifiers. The first and the second row show the average number of observations taken by OBSV (Fig. 3.21(a) - 3.21(b)) and by the oracle (Fig. 3.21(c) - 3.21(d)) over of all runs. The last row (Fig. 3.21(e) - 3.21(f)) shows the average in expected cost when OBSV is used for stopping. The dashed lines denote the $5th$- and $95th$-percentiles where a triangle denotes a value of zero.

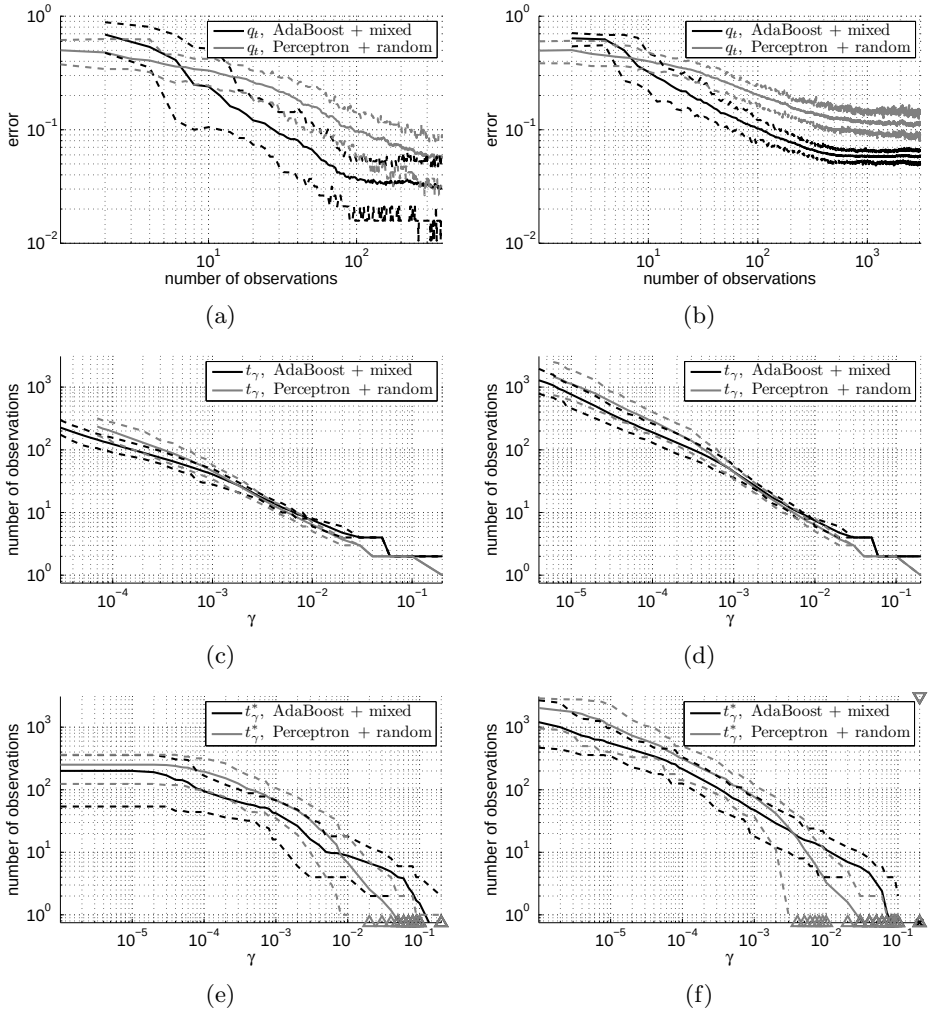


Figure 3.22: Results comparing **AdaBoost using mixed sampling to the Perceptron using random sampling** on the `wdbc` (left column) and the `spam` data (right column) as obtained from 60 runs of AdaBoost using 100 decision stumps as weak classifiers and of the Perceptron learning algorithm with 20 iterations over the current training set. The first row (Fig. 3.19(a) - 3.19(b)) shows the average in test error over all runs given these combinations of sampling and learning algorithms. The second and third row show the average number of observations taken by OBSV (Fig. 3.21(a) - 3.21(b)) and by the oracle (Fig. 3.21(c) - 3.21(d)) given these combinations. The dashed lines denote the $5th$- and $95th$-percentiles where a triangle denotes a value of zero.

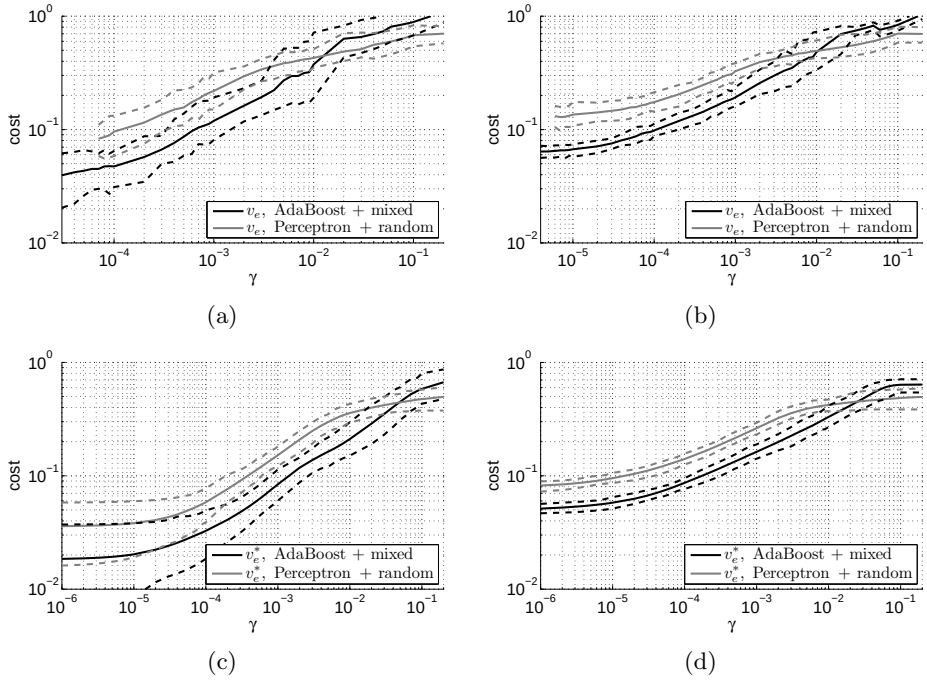


Figure 3.23: Results comparing **AdaBoost using mixed sampling to the Perceptron using random sampling** on the `wdbc` (left column) and the `spam` data (right column) as obtained from 60 runs of AdaBoost using 100 decision stumps as weak classifiers and of the Perceptron learning algorithm with 20 iterations over the current training set. The first row (Fig. 3.23(a) - 3.23(b)) shows the average in cost of OBSV over all runs given different combinations of sampling and learning algorithms. The second row (Fig. 3.23(c) - 3.23(d)) shows the average cost of the oracle given these combinations. The dashed lines denote the $5th$- and $95th$-percentiles.

the fraction between the expected monetary cost M_t for training the system up to time t and the monetary cost of a complete system failure,

$$\begin{aligned} \mathbf{E}[M_t \mid f_t, \mathscr{D}] &= M_R \cdot \mathbf{E}[R_t \mid f_t, \mathscr{D}] + M_L \cdot t \\ \Rightarrow \mathbf{E}[\underbrace{M_t/M_R}_{=:C_\gamma} \mid f_t, \mathscr{D}] &= \mathbf{E}[R_t \mid f_t, \mathscr{D}] + \underbrace{\frac{M_L}{M_R}}_{=:\gamma} \cdot t. \end{aligned}$$

Thus γ denotes the fraction between the monetary cost of labelling one image and the cost for a complete system failure. A complete failure may seem unrealistic in practice, since one may expect for example that the random hypothesis has an error of $r_0 = 1 - 1/n < 1$. Then one would rather choose

$$\gamma = \frac{M_L}{r_0 \cdot M_R}. \tag{3.36}$$

Below, we describe scenarios from practice where an image categorisation system has to distinguish between two different visual categories. Thereby we will calculate γ using reasonable assumptions about monetary costs in the application domain:

- *Toll Collection for anonymous customers.* Toll collection on highways involves vehicle dependent rates. Currently Austria's ASFINAG offers video-based road-taxation for registered customers. However the majority of anonymous customers still have to be served by humans. As there are many toll stations, that company might be interested in a system that is capable of categorising for example cars and buses based on image data to determine the correct toll, e.g. 5 € for cars and 10 € for buses. In this scenario a false classification of a bus leads to a deficit of 5 € whereat a misclassification of a car involves a reclamation process with a handling charge of e.g. 5 € for the company. Assuming that the expected traffic volume during the system's lifetime is about 10^6, the cost of a complete system failure is $M_R = 5 \cdot 10^6$ €. Now if labelling one image costs the company $M_L = 2$ €, according to Eq. 3.36 it would choose

$$\gamma = \frac{2\,€}{0.5 \cdot 5 \cdot 10^6\,€} = 8 \cdot 10^{-7}.$$

- *Vehicle dependent parking fees.* The municipality of a city is operating several parking-lots and may wish to charge vehicle dependent fees, e.g. for cars and motorbikes, without employing parking wardens. Using cameras put up at the barriers the task is now to train a classifier for categorising the images. Assume that the fee is 4 € for cars and only 2 € for motorbikes. In this scenario a false positive classification of a car would lead to a deficit of 2 €, while upon a false negative

classification the customer is likely to leave which causes also a deficit of 2€. Assuming that the municipality expects $5 \cdot 10^4$ customers during the system's lifetime we have $M_R = 2 \cdot 5 \cdot 10^4 € = 10^5 €$. Now if labelling one image costs the municipality $M_L = 2€$, according to Eq. 3.36 it would choose

$$\gamma = \frac{2€}{0.5 \cdot 10^5 €} = 4 \cdot 10^{-5}.$$

For the scenarios we assumed that both types of misclassification error incur the same monetary cost. Our model however is not limited to such cases since we can incorporate various misclassification costs into our cost function as follows.

Let $\mathbf{M} \in [0,1]^{|\mathcal{Y}| \times |\mathcal{Y}|}$, with $\sum_{i,j=1}^{|\mathcal{Y}|} M_{i,j} = 1$, be some normalised matrix for monetary misclassification costs, i.e. $M_{i,j}$ denotes the relative monetary costs for predicting class i instead of the true class j. Then the total cost of a hypothesis f_t at time t w.r.t. $\mathbf{M}$ can be written as

$$\begin{aligned}\mathbf{E}[C_\gamma \mid f_t, \mathscr{D}, \mathbf{M}] &= \mathbf{E}[R_t \mid f_t, \mathscr{D}, \mathbf{M}] + \gamma t \\ &= \gamma t + \int_{\mathcal{X}} \sum_{y \in \mathcal{Y}} p(y \mid x, \mathscr{D})\, M_{f_t(x),y}\, p(x \mid \mathscr{D})\, dx\end{aligned}$$

and we can apply OBSV to estimate the optimal stopping time. It is useful to note that there exist cost-sensitive learning algorithms to infer f_t for such learning problems. The interested reader may refer to [64, 63] for a recent review of cost-sensitive classification using boosting.

3.4.2 Experimental setup

The purpose of this section is to evaluate the performance of the combined framework given image data according to our scenarios. First we relate its performance to the oracle's given different sampling strategies. Then we compare the performance of these sampling strategies to each other. We plot stopping times and cost-curves for a range of values of γ utilising multiple runs of cross-validation over data sets from the `VOC06` database.

For the experiments we use 1097 images of cars, 354 of buses and 469 for motorbikes with the same features as described in Sec. 1.5.3. For each of the two resulting data sets, cars vs. buses and cars vs. motorbikes, we perform 5 runs of 2-fold stratified cross-validation. For each run, we split the data into a training set $\mathcal{D}$ and test set $\mathcal{D}_E$, the training set itself being split into a random sampling set $\mathcal{D}_R$ and active sampling set $\mathcal{D}_A$ whenever appropriate. To reduce the computational effort we query multiple labels within each iteration of OBSV. That is for *random* sampling we query 3 unlabelled examples from $\mathcal{D}_R$ without replacement. For *mixed sampling* we

actively query another 3 examples without replacement from $\mathcal{D}_A$, which are closest to the decision boundary of the current classifier.

As our main focus is optimal stopping we reduce the computational effort for learning each f_t by (a) using AdaBoost (with 100 decision stumps as weak classifiers) instead of LPBoost and (b) omitting geometric relations between features as done in Sec. 1.5.3[4].

3.4.3 Comparing OBSV to the baselines given different sampling strategies

We start by comparing the performance of OBSV given random and mixed sampling. For this purpose we examine its stopping times and costs when compared to (a) $\widehat{\text{OBSV}}$, (b) the naive stopping algorithm and (c) the oracle. Note that active sampling is omitted since the violation of the i.i.d. assumption prohibits its use for concrete scenarios such as those considered here (see Sec. 3.4.1).

Random sampling

First we evaluate the performance of OBSV given the random sampling strategy. For reference Fig. 3.24 shows the average test errors q_t for cars vs. buses and for cars vs. motorbikes over all runs. It becomes obvious that the error rates are too large for practical situations since the acceptance of the costumers would be very low. However the final errors reported here are almost the same as the error rates as observed for LPBoost using no geometric relations.

As shown by Fig. 3.24(c) - 3.24(d), $\widehat{\text{OBSV}}$ stops too early due to overfitting, while the naive stopping strategy is always late. The stopping times of OBSV however follow those of the oracle closely. It is important to note that we observed similar results on `wdbc` and `spam` (Sec. 3.3.1). Consequently the average ratio in cost for using OBSV instead of the oracle is lower by up to an order of magnitude when compared to the average ratio in cost of using $\widehat{\text{OBSV}}$ instead of the oracle (Fig. 3.24(e) - 3.24(f)). Furthermore for cars vs. motorbikes and $\gamma = 4 \cdot 10^{-5}$, as given by our second scenario, the additional costs of OBSV when compared to the oracle are bounded from above by a around 30%. For cars vs. buses the given dataset is too small to investigate reasonable values of $\gamma < 3 \cdot 10^{-5}$. However considering the available values, the additional costs of OBSV for cars vs. buses are about as good as that for cars vs. motorbikes.

[4]Since we have two-class problems and omit geometry, the decision stumps here can have both orientations, i.e. $f(x;\theta) = [x \leq \theta]$ or $f(x;\theta) = [x > \theta]$.

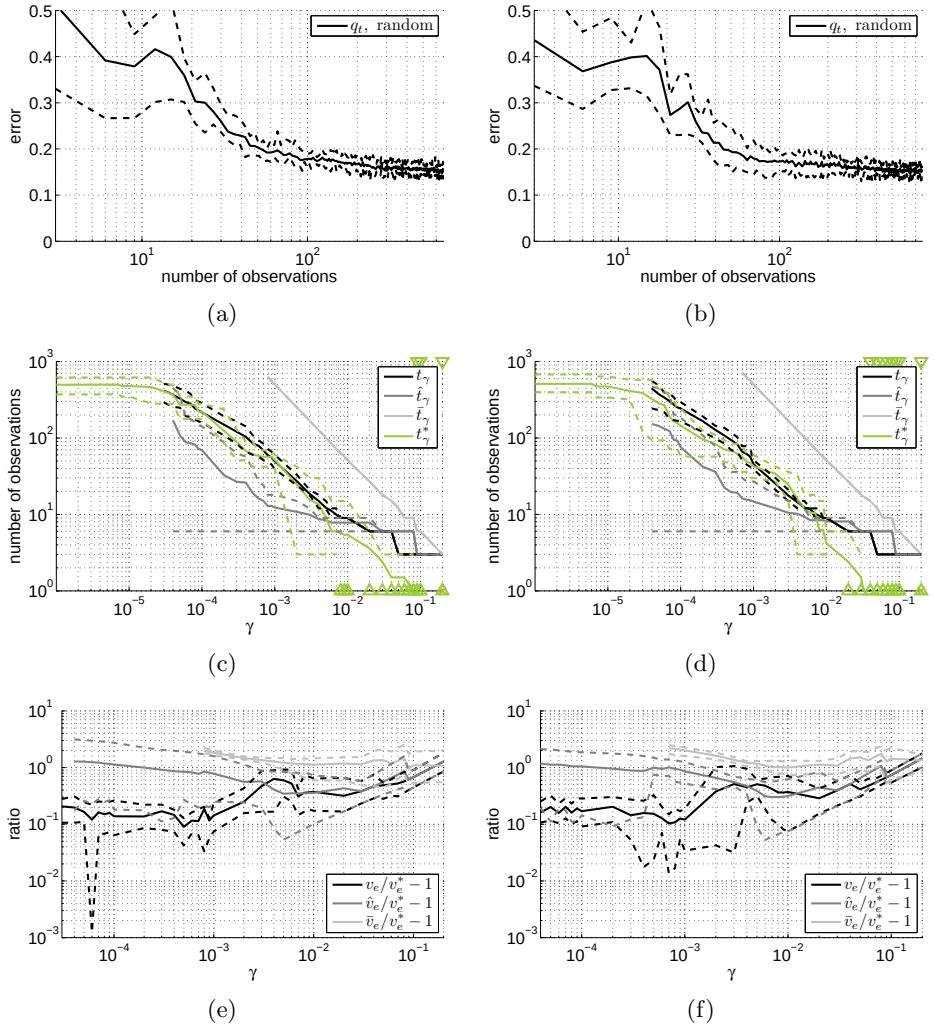


Figure 3.24: Results for **random sampling** on cars vs. buses (left column) and on cars vs. motorbikes (right column) as obtained from 10 runs of AdaBoost using 100 decision stumps as weak classifiers. The first row (Fig. 3.24(a) - 3.24(b)) shows the realised error as measured on the test set averaged over of all runs. The second row (Fig. 3.24(c) - 3.24(d)) shows the average of the expected number of observations taken by OBSV, $\widehat{\text{OBSV}}$, the naive stopping algorithm and the oracle. The third row (Fig. 3.24(e) - 3.24(f)) shows the average ratio in cost w.r.t. the oracle. The dashed lines denote the $10th$- and $90th$-percentiles where a triangle denotes a value of zero.

Mixed sampling

We perform the same analysis for mixed sampling. As a reference the test errors are shown in Fig. 3.25(a) - 3.25(b). Fig. 3.25(c) - 3.25(f) show that the stopping times and therefore the additional costs of OBSV are much better when compared to the other baselines. Furthermore we note that these results are very similar to those for random sampling. Therefore it remains questionable whether mixed sampling gives any advantage over random sampling. This will be investigated next.

3.4.4 Comparing different sampling strategies

As shown by Fig. 3.26(a) - 3.26(b) the test error curves given random sampling and given mixed sampling are almost identical. Consequently the stopping times of OBSV are also approximately the same as shown by Fig. 3.26(c) - 3.26(d). Considering Fig. 3.26(e) - 3.26(f) we note that the stopping times of the oracle also overlap strongly. Thus as shown by Fig. 3.27(a) - 3.27(b), the average in cost of both sampling strategies are almost identical when OBSV utilised for stopping. This corresponds to the average in cost for the oracle shown below (Fig. 3.27(c) - 3.27(d)).

3.5 Discussion

The results of this chapter showed that OBSV produces stopping times that are reasonably close to optimal. Specifically, we demonstrated that for the range of values for γ and the specific prior used here our model can adapt to artificial learning curves from various classes of convergence. Thereby it turned out that OBSV always outperformed the other baselines. We confirmed those results on real data from the `UCI` database and the `VOC06` database using various sampling strategies and learning algorithms. Perhaps the most interesting results for active learning practitioners are (a) the different stopping times and costs obtained from OBSV given significantly different learning curves and (b) the comparable results of random sampling w.r.t. the mixed or active strategy which sample along the current decision boundary. By combining both frameworks, image categorisation through boosting and OBSV, we provided a system that is capable of learning various classes while learning itself can be stopped according to a customisable cost measure. Although the error rates reported on images from `VOC06` would not be acceptable in practice, we believe that this is an adequate first step and deserves further attention.

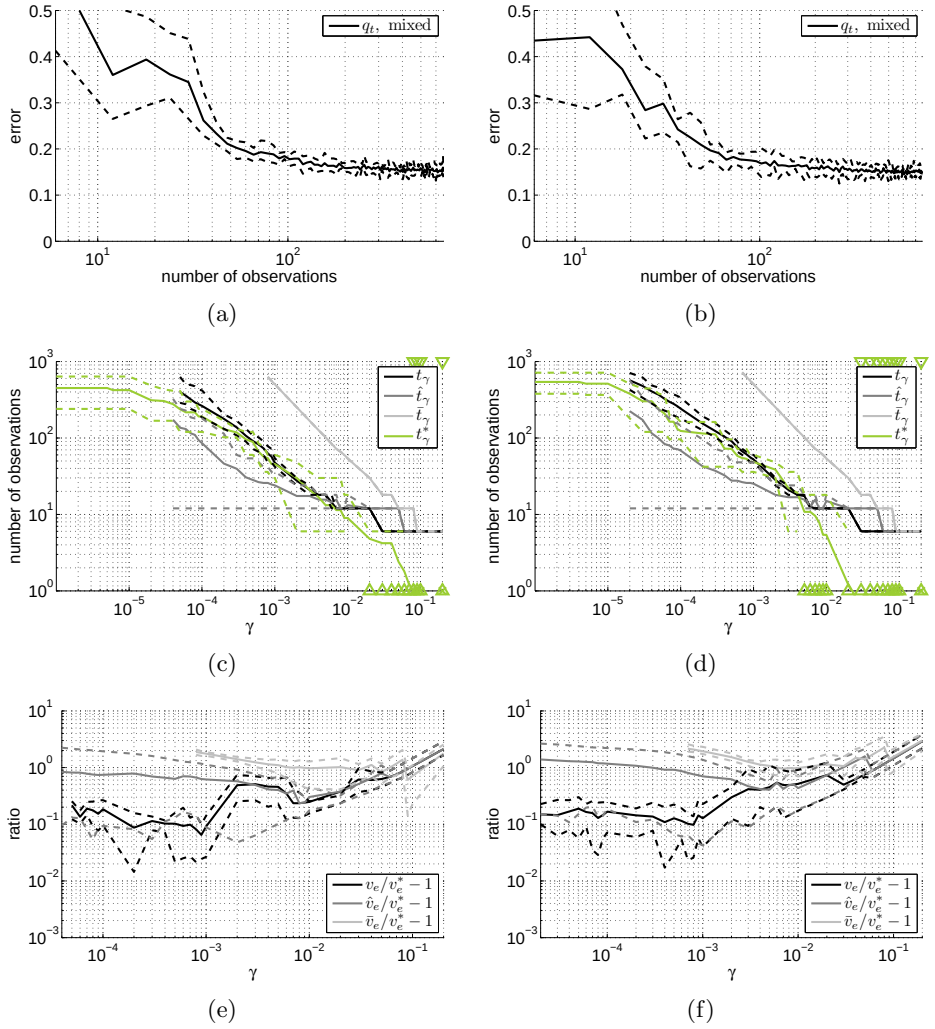


Figure 3.25: Results for **mixed sampling** on cars vs. buses (left column) and on cars vs. motorbikes (right column) as obtained from 10 runs of AdaBoost using 100 decision stumps as weak classifiers. The first row (Fig. 3.25(a) - 3.25(b)) shows the realised error as measured on the test set averaged over of all runs. The second row (Fig. 3.25(c) - 3.25(d)) shows the average of the expected number of observations taken by OBSV, $\widehat{\text{OBSV}}$, the naive stopping algorithm and the oracle. The third row (Fig. 3.25(e) - 3.25(f)) shows the average ratio in cost w.r.t. the oracle. The dashed lines denote the $10th$- and $90th$-percentiles where a triangle denotes a value of zero.

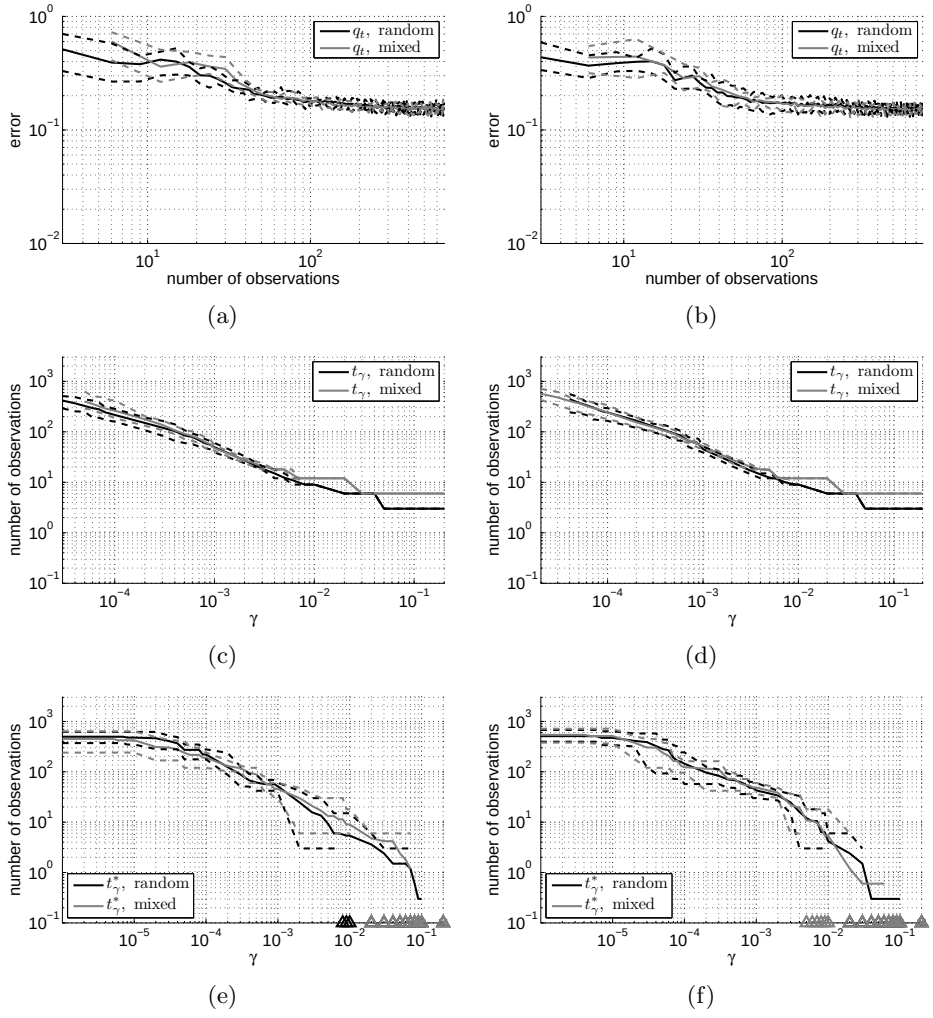


Figure 3.26: Results comparing **random and mixed sampling** on cars vs. buses (left column) and on cars vs. motorbikes (right column) as obtained from 10 runs of **AdaBoost** using 100 decision stumps as weak classifiers. The first row (Fig. 3.26(a) - 3.26(b)) show the average in test error over of all runs. The second and the third rows show the average number of observations taken by OBSV (Fig. 3.26(c) - 3.26(d)) and by the oracle (Fig. 3.26(e) - 3.26(f)) over of all runs. The dashed lines denote the $10th$- and $90th$-percentiles where a triangle denotes a value of zero.

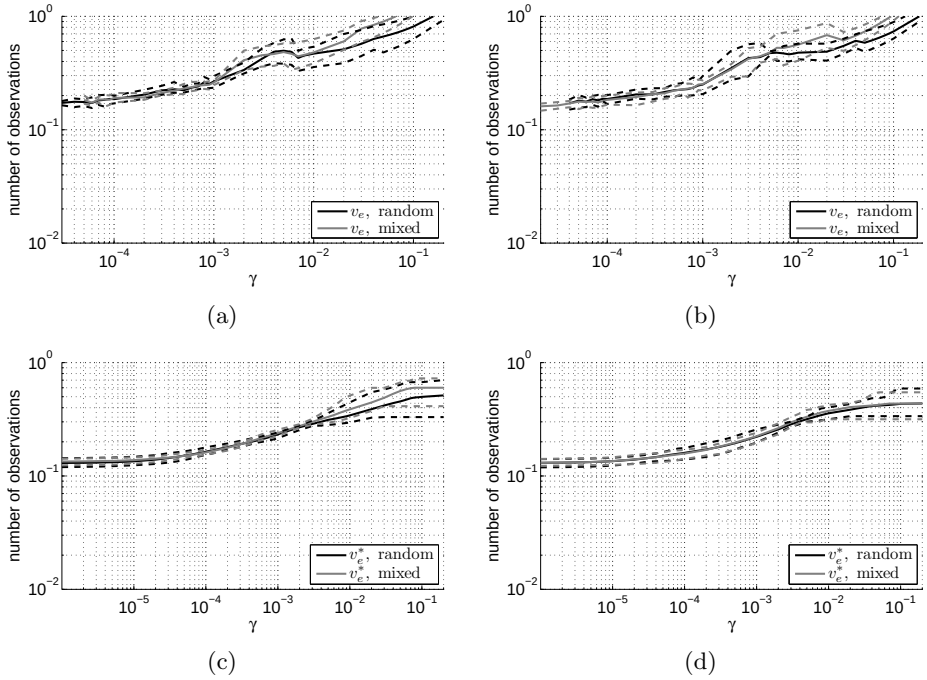


Figure 3.27: Results comparing **random and mixed sampling** on cars vs. buses (left column) and on cars vs. motorbikes (right column) as obtained from 10 runs of **AdaBoost** using 100 decision stumps as weak classifiers. The first row row (Fig. 3.27(a) - 3.27(b)) shows the average in expected cost over all runs when OBSV is used for stopping. The second row (Fig. 3.27(c) - 3.27(d)) shows the average in cost over all runs given the oracle. The dashed lines denote the $10th$- and $90th$-percentiles.

Chapter 4

Conclusion

The first chapter presented extensions to the image catgorisation framework of Opelt et al. by (a) incorporating geometric relations between features into the weak learner as proposed in [3] and (b) providing a weight optimisation method to combine pairwise classifiers for multiclass classification as proposed in [4]. The presented results on `Xerox` showed that our method outperforms the bag-of-keypoints approach of Csurka et al. and that the geometric relations learned are reasonable. Furthermore, the results at the PASCAL VOC Challenge 2006 showed that, even without geometry, we achieve reasonable results when compared to other methods although we (a) made little effort in tuning target-specific feature extractors, the number of clusters and so on, and (b) utilise the predictions of *only one* multiclass classifier to distinguish among *all* classes. Thus, the most interesting result for practitioners may be the intrinsic flexibility of our framework given different visual categories.

The last chapters discussed the interplay between a well-defined cost function, stopping algorithms and objective evaluation criteria and their relation to active learning. Specifically, we have argued that (a) learning when labels are costly is essentially a stopping problem, (b) it is possible to use optimal stopping procedures based on a suitable cost function, (c) the goal of active learning algorithms could also be represented by this cost function, (d) metrics on this cost function should be used to evaluate performance and finally that, (e) the stopping problem cannot be separately considered from either the cost function or the evaluation. To our current knowledge, these issues have not been sufficiently addressed in previous work.

For this reason, we have proposed a suitable cost function and presented a practical stopping algorithm which aims to be optimal with respect to this cost. Experiments with this algorithm for a specific prior show that it suffers only small loss compared to the optimal stopping time and is certainly a step forward from ad hoc stopping rules. Perhaps the most interesting results for practitioners are that (a) given the data set considered

the performance of active sampling along the current decision boundary is not superior compared to random sampling and (b) the combination of our image categorisation framework and OBSV according to concrete scenarios from practice yields reasonable stopping times and therefore costs.

On the other hand, while the presented stopping algorithm is an adequate first step, its combination with active learning is not perfectly straightforward since the balance between active and uniform sampling is a hyperparameter which is not obvious how to set.[1] An alternative is to use model-specific stopping methods. This could be done if we restrict ourselves to probabilistic classifiers, as for example in [14]; this way we may be able to simultaneously perform optimal example selection and stopping. If such a classifier is not available for the problem at hand, then judicious use of frequentist techniques such as bootstrapping [23] may provide a sufficiently good alternative for estimating probabilities. Such an approach was advocated by [56] in order to optimally select examples; however in our case we could extend this to optimal stopping. Briefly, this can be done as follows.

Let our belief at time t be ξ_t, such that for any point $x \in \mathcal{X}$, we have a distribution of probabilities $\mathbf{P}(y \mid x, \xi_t)$ over $\mathcal{Y}$. For example when using boosting algorithms such as given by Alg. 1, 2 (see Sec. 1.3.2), according to [30] we could calculate

$$\mathbf{P}(y \mid x, \xi_t) = \sum_{j: f_j(x)=y} a_j,$$

where a_j is the weight of the weak classifier f_j within the strong classifier at time t and we normalized $\sum_{j=1}^{J} a_j = 1$ if necessary. We may now calculate this over the whole dataset to estimate the realised generalisation error as the *expected error given the empirical data distribution and our classifier*

$$\mathbf{E}_{\mathcal{D}}(v_t \mid \xi_t) = \frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} [1 - \arg\max_y \mathbf{P}(y_i = y \mid x_i, \xi_t)]. \tag{4.1}$$

We can now calculate (4.1) for each of the different possible classes. So we calculate the *expected error on the empirical data distribution if we create a new classifier from ξ_t by adding example i* as

$$\mathbf{E}_{\mathcal{D}}(v_t \mid x_i, \xi_t) = \sum_{y \in \mathcal{Y}} \mathbf{P}(y_i = y \mid x_i, \xi_t)\, \mathbf{E}_{\mathcal{D}}(v_t \mid x_i, y_i = y, \xi_t), \tag{4.2}$$

where we note that $\mathbf{P}(y_i = y \mid x_i, \xi_t)$ is just the probability of example i having label y according to our current belief, ξ_t. Furthermore, $\mathbf{E}_{\mathcal{D}}(v_t \mid x_i, y_i = y, \xi_t)$ results from calculating (4.1) using the classifier resulting from ξ_t and the added example i with label y. Then $\mathbf{E}_{\mathcal{D}}(v_t, \xi_t) - \mathbf{E}_{\mathcal{D}}(v_t \mid x_i, \xi_t)$ will be the expected gain from using i to train. The (subjectively) optimal

[1] In this work, the active and uniform sampling rates were equal.

one-step stopping algorithm is as follows: Let $i^* = \arg\min_i \mathbf{E}_{\mathcal{D}}(v_t \mid x_i, \xi_t)$. Stop if $\mathbf{E}_{\mathcal{D}}(v_t \mid \xi_t) - \mathbf{E}_{\mathcal{D}}(v_t \mid x_{i^*}, \xi_t) < \gamma$.

A particular difficulty in the presented framework, and to some extent also in the field of active learning, is the choice of hyperparameters for the classifiers themselves. For Bayesian models it is possible to select those that maximise the marginal likelihood.[2] One could alternatively maintain a set of models with different hyperparameter choices and separate convergence estimates. In that case, training would stop when there are no models for which the expected gain is larger than the cost of acquiring another label. Even this strategy, however, is problematic in the active learning framework, where each model may choose to query a different example's label. Thus, the question of hyperparameter selection remains open and should be addressed it in future work.

Finally, we hope that the presented exposition will increase awareness of optimal stopping and evaluation issues in the active learning community, lead to commonly agreed standards for the evaluation of active learning algorithms, or perhaps even encourage the development of example selection methods incorporating the notions of optimality suggested in this work.

[2]Other approaches require the use of techniques such as cross-validation, which creates further complications.

Appendix A

Supplemental learning algorithms

A.1 The Perceptron learning algorithm

Algorithm 6 Perceptron

Given a data set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, +1\}$ and a maximum number of iterations J,

initialise $\mathbf{w} \leftarrow \mathbf{0} \in \mathbb{R}^{1\times d},\ b \leftarrow 0$,
for $j = 1 : J$ **do**
  for $i = 1 : m$ **do**
    calculate prediction for the current example:
    if $\mathbf{w}\mathbf{x}_i + b \geq 0$ **then** $y_i' \leftarrow 1$ **else** $y_i' \leftarrow -1$ **end**
    update upon an error:
    if $y_i' \neq y_i$ **then**
      $\mathbf{w} \leftarrow \mathbf{w} + y_i\mathbf{x}_i^T,\ b \leftarrow b + y_i$
    end if
  end for
end for
return $f(\mathbf{x}; \mathbf{w}, b) = \begin{cases} 1, & \mathbf{w}\mathbf{x} + b \geq 0 \\ -1, & \text{else} \end{cases}$

Bibliography

[1] Naoki Abe and Hiroshi Mamitsuka. Query learning strategies using boosting and bagging. In *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*, pages 1–9, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[2] Dana Angluin. Queries and concept learning. *Mach. Learn.*, 2(4):319–342, 1988.

[3] M. Antenreiter. Objekterkennung durch Lernen der relevanten Geometrie von Bezugspunkten, 2005. Diploma thesis.

[4] Martin Antenreiter, Christian Savu-Krohn, and Peter Auer. Visual classification of images by learning geometric appearances through boosting. In *ANNPR*, pages 233–243, 2006.

[5] A. Asuncion and D.J. Newman. UCI machine learning repository. http://mlearn.ics.uci.edu/MLRepository.html, 2007.

[6] Maria-Florina Balcan, Alina Beygelzimer, and John Langford. Agnostic active learning. In William W. Cohen and Andrew Moore, editors, *23rd International Conference on Machine Learning*, pages 65–72. ACM, 2006.

[7] Maria-Florina Balcan, Steve Hanneke, and Jennifer Wortman. The true sample complexity of active learning. In *COLT*, pages 45–56, 2008.

[8] Samy Bengio, Johnny Mariéthoz, and Mikaela Keller. The expected performance curve. In *International Conference on Machine Learning ICML'07*, 2005.

[9] Kristin P. Bennett, Ayhan Demiriz, and John Shawe-Taylor. A column generation algorithm for boosting. In *Proc. 17th International Conf. on Machine Learning*, pages 65–72. Morgan Kaufmann, San Francisco, CA, 2000.

[10] Leo Breiman. Prediction games and arcing algorithms. *Neural Comput.*, 11(7):1493–1517, 1999.

[11] M.C. Burl, T.K. Leung, and P. Perona. Recognition of planar object classes. In *Proceedings of the 1996 Conference on Computer Vision and Pattern Recognition (CVPR '96)*, pages 223–230. IEEE Computer Society, 1996.

[12] Michael C. Burl, Markus Weber, and Pietro Perona. A probabilistic approach to object recognition using local photometry and global geometry. In *ECCV '98: Proceedings of the 5th European Conference on Computer Vision-Volume II*, volume 1407, pages 628–641. Springer-Verlag, 1998.

[13] C. Campbell, N. Cristianini, and Alex Smola. Query learning with large margin classifiers. In *Proceedings of the 17th International Conference on Machine Learning, ICML-2000*, pages 111–118, 2000.

[14] David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. Active learning with statistical models. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 705–712. The MIT Press, 1995.

[15] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:603–619, 2002.

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

[17] David Crandall, Pedro F. Felzenszwalb, and Daniel P. Huttenlocher. Spatial priors for part-based recognition using statistical models. In *CVPR (1)*, pages 10–17. IEEE Computer Society, 2005.

[18] Gabriela Csurka, Cedric Bray, Christopher Dance, and Lixin Fan. Visual categorization with bags of keypoints. In *European Conference on Computer Vision, ECCV'04*, Prague, Czech Republic, May 2004. Dataset available upon request.

[19] Sanjoy Dasgupta, Daniel Hsu, and Claire Monteleoni. A general agnostic active learning algorithm. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 353–360. MIT Press, Cambridge, MA, 2008.

[20] Morris H. DeGroot. *Optimal Statistical Decisions*. John Wiley & Sons, 1970. Republished in 2004.

[21] Ayhan Demiriz, Kristin P. Bennett, and John Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46(1–3):225–254, 2002.

[22] Christos Dimitrakakis and Christian Savu-Krohn. Cost-minimising strategies for data labelling: Optimal stopping and active learning. In *FoIKS*, pages 96–111, 2008.

[23] Bradley Efron and Robert J. Tibshirani. *An Introduction to the Bootstrap*, volume 57 of *Monographs on Statistics & Applied Probability*. Chapmann & Hall, November 1993.

[24] 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.

[25] Jason Farquhar, Sandor Szedmak, Hongying Meng, and John Shawe-Taylor. Improving bag-of-keypoints image categorisation: Generative models and pdf-kernels. Technical report, Image Speech and Intelligent Systems, Department of Electronics and Computer Science, University of Southampton, 2005. Third place at the PASCAL Visual Objects Classes Challenge, Southampton, 2005.

[26] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *Int. J. Comput. Vision*, 59(2):167–181, 2004.

[27] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[28] R. Fergus, P. Perona, and A. Zisserman. Weakly supervised scale-invariant learning of models for visual recognition. *Int. J. Comput. Vision*, 71(3):273–303, 2007.

[29] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *European Conference on Computational Learning Theory*, pages 23–37, 1995.

[30] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, 1997.

[31] Michael Fussenegger, Andreas Opelt, Axel Pinz, and Peter Auer. Object recognition using segmentation for feature detection. In *ICPR (3)*, pages 41–44, 2004.

[32] R. Gonzalez and R. Woods. *Digital Image Processing*. Prentice Hall, 2007.

[33] Luc J. Van Gool, Theo Moons, and Dorin Ungureanu. Affine / photometric invariants for planar intensity patterns. In *ECCV'96: Proceedings of the 4th European Conference on Computer Vision-Volume I*, pages 642–651. Springer-Verlag, 1996.

[34] R. M. Haralick. Statistical and structural approaches to texture. 67(5):786–804, 1979.

[35] Thorsten Joachims. Making large-scale support vector machine learning practical. In *Advances in kernel methods: support vector learning*, pages 169–184. MIT Press, Cambridge, MA, USA, 1999.

[36] Matti Kääriäinen. Active learning in the non-realizable case. In *Proceedings of the 17th International Conference on Algorithmic Learning Theory*, 2006.

[37] Timor Kadir and Michael Brady. Saliency, scale and image description. *International Journal of Computer Vision*, 45(2):83–105, 2001.

[38] Christine Körner and Stefan Wrobel. Multi-class ensemble-based active learning. In *Proceedings of ECML-2006*, volume 4212 of *LNAI*, pages 687–694, 2006.

[39] Bastian Leibe, Ales Leonardis, and Bernt Schiele. Combined object categorization and segmentation with an implicit shape model. In *ECCV workshop on statistical learning in computer vision*, pages 17–32, 2004.

[40] D.G. Lowe. Object recognition from local scale-invariant features. In *Seventh International Conference on Computer Vision*, pages 1150–1157, 1999.

[41] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[42] S. Maitra. Moment invariants. 67(4):679–699, 1979.

[43] Prem Melville and Raymond J. Mooney. Diverse ensembles for active learning. In *Proceedings of the 21st International Conference on Machine Learning, ICML-2004*, pages 584–591, 2004.

[44] K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. In *International Conference on Computer Vision*, pages 525–531, 2001.

[45] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *European Conference on Computer Vision*, pages 128–141, 2002.

[46] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. In *CVPR (2)*, pages 257–263, 2003.

[47] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(10):1615–1630, 2005.

[48] Thomas Mitchell. *Machine Learning*. McGraw-Hill Education (ISE Editions), 1997.

[49] Andreas Opelt, Michael Fussenegger, Axel Pinz, and Peter Auer. Weak hypotheses and boosting for generic object detection and recognition. In *Proc. of the 8th European Conference on Computer Vision (ECCV)*, volume 2, pages 71–84, 2004.

[50] Andreas Opelt, Michael Fussenegger, Axel Pinz, and Peter Auer. Generic object recognition with boosting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(3):416–431, March 2006.

[51] Emanuel Parzen. On estimation of a probability density function and mode. *Annals of Mathematical Statistics*, 33(3):1065–1076, 1962.

[52] Axel Pinz. Object categorization. *Foundations and Trends in Computer Graphics and Vision*, 1(4), 2005.

[53] J. R. Quinlan. Boosting first-order learning. In *Lecture Notes in Computer Science*, pages 143–155. Springer, 1996.

[54] G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for adaboost. In *Machine Learning*, pages 287–320, 2001.

[55] G. Rätsch, B. Schölkopf, A. Smola, K.-R. Müller, T. Onoda, and S. Mika. ν-Arc: Ensemble learning in the presence of outliers. In D.A. Cohn M.S. Kearns, S.A. Solla, editor, *Advances in Neural Information Processing Systems 12*. MIT Press, 2000.

[56] Nicholas Roy and Andrew McCallum. Toward optimal active learning through sampling estimation of error reduction. In *Proc. 18th International Conf. on Machine Learning*, pages 441–448. Morgan Kaufmann, San Francisco, CA, 2001.

[57] Maytal Saar-Tsechansky and Foster Provost. Active learning for class probability estimation and ranking. In *Proceedings of the 17th international joint conference on articial intelligence - IJCAI 2001*, pages 911–920, 2001.

[58] Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: a new explanation for the effectiveness of voting methods. *Annals of statistics*, 26(5):1651–1686, 1998.

[59] Robert E. Schapire and Yoram Singer. Improved boosting using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

[60] Cordelia Schmid, Roger Mohr, and Christian Bauckhage. Comparing and evaluating interest points. In *ICCV 1998*, pages 230–235, 1998.

[61] Cordelia Schmid, Roger Mohr, and Christian Bauckhage. Evaluation of interest point detectors. *International Journal of Computer Vision*, 37(2):151–172, 2000.

[62] G. Schohn and D. Cohn. Less is more: Active learning with support vector machines. In *Proceedings of the 17th International Conference on Machine Learning, ICML-2000*, pages 839–846, 2000.

[63] Yanmin Sun. *Cost-Sensitive Boosting for Classication of Imbalanced Data*. PhD thesis, University of Waterloo, Canada, 2007.

[64] Yanmin Sun, Mohamed S. Kamel, Andrew K. C. Wong, and Yang Wang. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, 40(12):3358–3378, 2007.

[65] Vladimir Vapnik. *The nature of statistical learning theory*. Springer, New York, 1995.

[66] Paul A. Viola and Michael J. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR (1)*, pages 511–518. IEEE Computer Society, 2001.

Affidavit:
I declare in lieu of oath, that I wrote this thesis and performed the associated research myself, using only literature cited in this volume.

Christian Savu-Krohn