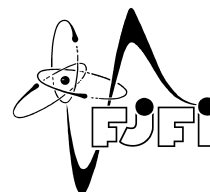




CZECH TECHNICAL UNIVERSITY IN
PRAGUE
Faculty of Nuclear Sciences and
Physical Engineering



Finding Practical Robust Classifiers

Hledání praktických robustních klasifikátorů

Master's Thesis

| | |
|----------------|-------------------------------|
| Author: | Bc. Dominik Šepák |
| Supervisor: | doc. Ing. Tomáš Pevný, Ph. D. |
| Consultant: | Mgr. Lukáš Adam, Ph. D |
| Academic year: | 2021/2022 |

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: Bc. Dominik Šepák
Studijní program: Aplikace přírodních věd
Studijní obor: Matematická informatika
Název práce (česky): Hledání praktických robustních klasifikátorů
Název práce (anglicky): Finding practical robust classifiers

Pokyny pro vypracování:

- 1) Nastudujte základní pojmy robustní klasifikace.
- 2) Představte a naimplementujte některé metody robustní klasifikace.
- 3) Vyberte vhodný problém, stručně ho představte a demonstруйте na něm implementované metody.
- 4) Porovnejte výsledky těchto metod na daném problému.



Doporučená literatura:

- 1) M. Arjovsky, et al., Invariant Risk Minimization. arXiv preprint arXiv:1907.02893, 2019.
- 2) E. Rosenfeld, P. Ravikumar, A. Ristestki, The risks of invariant risk minimization. arXiv preprint arXiv:2010.05761, 2020.
- 3) A. Rame, C. Dancette, M. Cord, Fishr: Invariant Gradient Variances for Out-of-distribution Generalization. arXiv preprint arXiv:2109.02934, 2021.
- 4) Q. Giboulot, et al., Effects and solutions of cover-source mismatch in image steganalysis. Signal Processing: Image Communication 86, 2020.

Jméno a pracoviště vedoucího diplomové práce:

doc. Ing. Tomáš Pevný, Ph.D.

Centrum umělé inteligence FEL ČVUT v Praze, Karlovo náměstí 13, 120 00 Praha 2

Jméno a pracoviště konzultanta:

Mgr. Lukáš Adam, Ph.D.

Centrum umělé inteligence FEL ČVUT v Praze, Karlovo náměstí 13, 120 00 Praha 2

Datum zadání diplomové práce: 31.10.2021

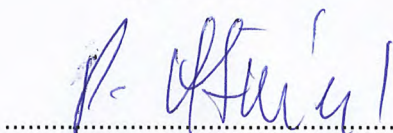
Datum odevzdání diplomové práce: 2.5.2022

Doba platnosti zadání je dva roky od data zadání.

V Praze dne 01.11.2021



garant oboru



vedoucí katedry





děkan

Acknowledgment:

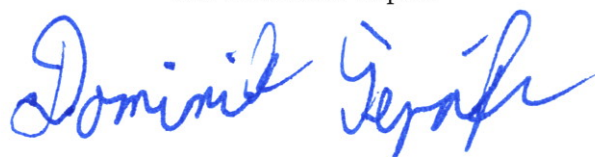
I would like to thank the supervisor of this thesis, doc. Ing. Tomáš Pevný, for his expert guidance of this thesis and his advice and comments on the conducted experiments and the text of this thesis. I would also like to thank the consultant of this thesis, Mgr. Lukáš Adam, Ph. D., for the many factual comments that significantly improved this thesis.

Čestné prohlášení:

Prohlašuji, že jsem tuto práci vypracoval samostatně a uvedl jsem všechnu použitou literaturu.

V Praze dne 2. května 2022

Bc. Dominik Šepák



Název práce:

Hledání praktických robustních klasifikátorů

Autor: Bc. Dominik Šepák

Obor: Matematická informatika

Druh práce: Diplomová práce

Vedoucí práce: doc. Ing. Tomáš Pevný, Ph. D., České vysoké učení technické v Praze, Fakulta elektrotechnická, Resslova 207/9, Praha

Konzultant: Mgr. Lukáš Adam, Ph. D., České vysoké učení technické v Praze, Fakulta elektrotechnická, Resslova 207/9, Praha

Abstrakt: V rámci této práce je představena studie některých metod strojového učení s učitelem a měř úspěšnosti predikce využívajících strukturovaných dat ve tvaru prostředí s různými data generujícími pravděpodobnostními distribucemi. V průběhu textu jsou také vysvětleny základní pojmy strojového učení s učitelem. Metody jsou následně prakticky porovnány na problému zvaném neshoda zdroje krytí (cover source mismatch) v obrazové steganografii.

Klíčová slova: cover source mismatch, robustní strojové učení, steganografie, strojové učení s učitelem

Title:

Finding Practical Robust Classifiers

Author: Bc. Dominik Šepák

Abstract: This work presents a study of some of the supervised machine learning methods and measures of prediction accuracy utilizing data in the structured form of environments with different data generating probability distributions. Basic concepts of supervised machine learning are explained throughout the text. The practical performance of the methods is then evaluated on the problem of cover source mismatch in image steganography.

Key words: cover source mismatch, robust learning, steganography, supervised machine learning

Contents

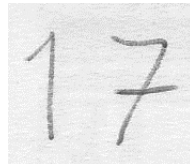
| | | |
|----------|---|-----------|
| 1 | Introduction | 11 |
| 2 | Introduction to Supervised Machine Learning | 15 |
| 2.1 | The Theory Behind Empirical Risk Minimization | 20 |
| 3 | Incorporating Environments Into Prediction Tasks | 25 |
| 3.1 | Measures of Quality of the Predictor | 26 |
| 3.2 | Environments in Supervised Machine Learning Methods | 28 |
| 3.2.1 | Clairvoyant Method | 28 |
| 3.2.2 | Mixture of Experts, Atomistic Approach | 29 |
| 3.2.3 | Robust Learning | 29 |
| 3.2.4 | Domain Adaptation | 32 |
| 4 | Application in Image Steganography | 39 |
| 4.1 | Short Introduction to Steganography | 39 |
| 4.1.1 | Steganalysis and Cover Source Mismatch | 41 |
| 4.2 | Experimental Details | 43 |
| 4.2.1 | Data | 43 |
| 4.2.2 | Methods | 44 |
| 4.3 | Experimental Results | 48 |
| 5 | Conclusion | 53 |
| A | Full Tables From the Experimental Section | 60 |

Chapter 1

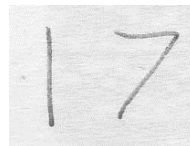
Introduction

Supervised machine learning is an art which, for data in the form (x, y) distributed according to probability distribution P^{all} , tries to find a function f (called predictor), which can estimate y from x . The predictor is practically created using a finite dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n_d}$ of samples sampled i.i.d. from P^{all} . However, in many applications, the predictor f can face samples from different probability distribution than P^{all} . The quality of the estimate of y can decrease in such cases. This problem is encountered in many practical applications of supervised machine learning algorithms, and we call it the problem of *robustness* of supervised machine learning algorithms to the change of the underlying probability distribution of the samples.

Let us illustrate the problem by an example. Consider the task of handwritten digit recognition using a supervised machine learning algorithm. When obtaining data to solve this task, we can invite many volunteers and let them each fill in a sample sheet, on which every participant writes digits in annotated fields prescribing which digit should be written in the respective field. The input of the predictor is then the image of the digit, and the desired output is an integer corresponding to the annotation of the field.



(a) Europe



(b) US

Figure 1.1: Different styles of writing the digits 1 and 7 between Europe and the United States [1]. Notice the similarity between the European 1 and American 7.

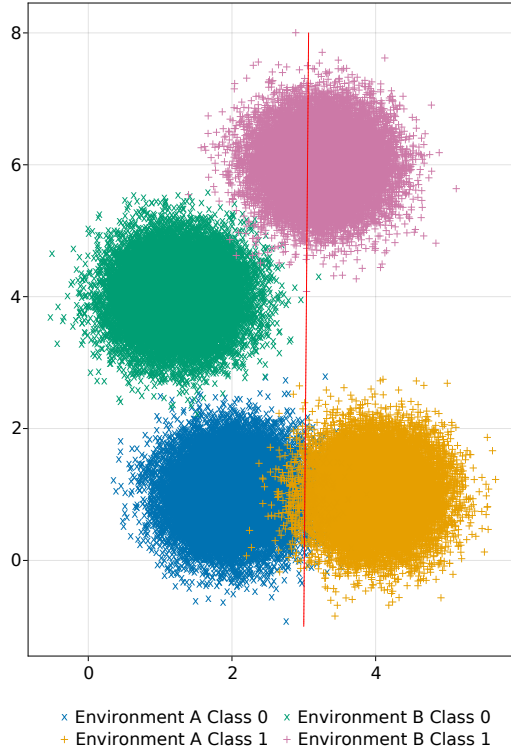


Figure 1.2: Example of two environments, A and B. Each environment consists of two classes which are to be recognised. The red line separates well the classes in the environment A and is optimal in the environment, but when used in environment B it gives significantly suboptimal results. The classes are sampled from normal distributions with equal covariance matrices and different means. There are 20000 samples in each class.

We could notice that each instance of the handwritten digit has its characteristics, making it possible to recognize it by other people. On the other hand, each participant has his/her specific style of writing, which makes his/her digits unique and distinguishable from other participants' digits. Sometimes it is possible to say who wrote the digits just from their look. Therefore, we can say that each participant forms an *environment* and that in each environment, the digits are drawn from slightly different *environment-specific probability distributions*. Suppose we create our predictor only on samples from one participant. In that case, it will likely fail when faced with digits from someone else – the example of this can be image 1.1, where a predictor created on the digits from a European participant could recognize American digits 7 as digits 1. Optimally, we would like the predictor to filter out the effect of the environment and learn only the useful characteristics of the digits.

To illustrate why such failure could happen, see the figure 1.2. The results are significantly suboptimal if we try to classify samples from environment B using only information from environment A. We will return to this example in later sections of the thesis.

To show that the problem of the robustness to the change of an environ-

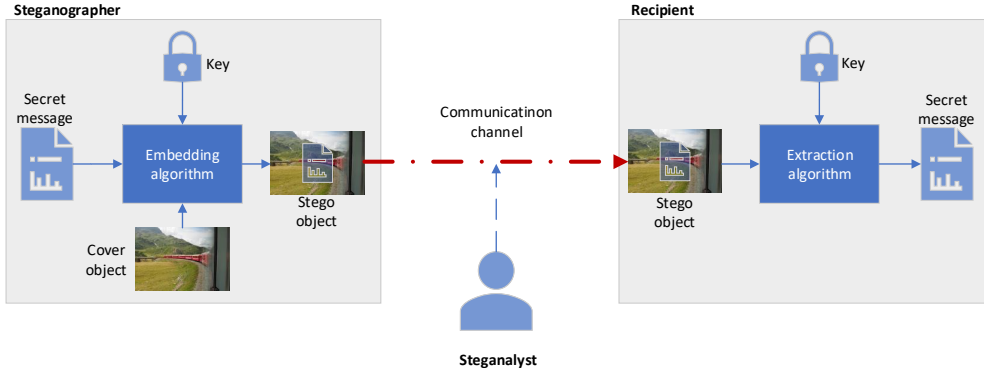


Figure 1.3: Communication scheme in image steganography.

ment is not artificial, we will introduce the reader to the problem of *cover source mismatch* [2] in image steganography. Steganography aims to communicate a secret message through an overt channel without the communication being detected. In the case of image steganography, such a channel has a form of images. Steganalysis then aims to detect the presence of the hidden secret message. Researchers created the steganographic algorithms (algorithms that hide the secret message in innocently looking data, for example, in an image from vacation) following the Kerckhoffs's principle [3], which states that the security of the algorithm must rely only on the secret key and it must be assumed that all parameters of the communication except the secret key can be known to the adversary. Unfortunately, it is a common practice that the same assumptions are made when designing the tools for steganalysis – it is assumed that the steganalyst has perfect knowledge about the steganographic algorithm, probability distribution of the *cover objects* (benign data, which is later used to hide the message, i.e., a set of images), and the length of the message. This assumption on the steganalyst's knowledge is highly unrealistic in practical application, and if the steganalyst's assumed probability distribution of the cover objects differs from the actual probability distribution of the cover objects, the performance of the algorithms used for steganalysis decreases, as shown in [2]. This problem is called *cover source mismatch*.

The cover source mismatch can be formulated using the notion of environments and change of the data generating probability distributions. Each cover source forms an environment with its specific data generating probability distribution. The goal is to find a steganalytic predictor robust to the change of the environment (cover source, data-generating probability distribution). The predictor then must detect the useful signal (the hidden

message) while unaffected by the environment-specific signal in the communication channel (i.e., the contents of the cover image).

Some solutions were proposed to mitigate the cover source mismatch [4] [5] [6]. However, the problem is mostly overlooked by the community, which is why we find it interesting. We will use it as a benchmark to demonstrate the algorithms presented in this thesis and measure their performance on the problem. We hope that the thesis will bring original results on this problem.

The outline of the work is following. In chapter 2, we present the reader with basic concepts of supervised machine learning and the concept of empirical risk minimization. We also show why the concept is appealing from the theoretical and practical points of view. Then we point out the weakness of this concept when applied to the problem mentioned above. In chapter 3, we will study the notion of environment and different measures of quality of prediction and methods utilizing the environments. In chapter 4 we will provide a short introduction to image steganography and demonstrate the concepts from the previous chapter on the problem of cover source mismatch in image steganography.

Chapter 2

Introduction to Supervised Machine Learning

Recall the handwritten digit recognition task. In this task, we would like an instance of a handwritten digit to form the input of function f , and we would like the function to output which digit it is reliably. We can restrict ourselves to 64×64 pixels grayscale image of each digit. We will then transform the image into a vector by concatenating the rows of the image, obtaining a real-valued vector with 4096 components. This is the actual *input* x of the function f . Throughout the thesis, we will assume that the input is an n -dimensional real-valued vector from some set $\mathcal{X} \subset \mathbb{R}^n$.

We will further assume that each input x can be unequivocally given a *label* y , which is a machine-readable annotation of the data. In the handwritten digit recognition task, the label would clearly state which digit (zero to nine) is captured in the input. Formally, the label is an m -dimensional vector from some set $\mathcal{Y} \subset \mathbb{R}^m$.

The tuple (x, y) of the input and the corresponding label is called *sample*. We assume that we can use a set of samples to find the function f . Machine learning problems with data in the form of inputs annotated by labels are called *supervised* [7]. When we find the function f and deploy it into its application, we assume that it will be provided only the input data x , and we would like it to output the correct label y (or at least a good estimate of the label).

We further assume that the samples are sampled from some probability distribution P^{all} over $\mathcal{X} \times \mathcal{Y}$. We can summarize and detail **our goal** as to find a function f from some set of functions \mathcal{F} , which, for any sample (x, y) sampled from P^{all} , takes x as an input and outputs a good **estimate** $\hat{y} = f(x)$ of the label y . We will call f *predictor* and \mathcal{F} *hypothesis space* or the *set of possible predictors*. We want the estimate to be as good as possible. The

quality of the estimate is measured by a real-valued *loss function* ℓ , which takes the estimate \hat{y} and label y as its inputs and outputs a value, which tells us how well \hat{y} estimates y . Good assumptions on the loss function are that the output is non-negative and the lower is the output, the closer is \hat{y} to y . Ideally, we would like the loss function to be zero for matching input arguments.

For discrete \mathcal{Y} , the image of f is often an open superset of \mathcal{Y} . This allows to continuously decrease the distance of \hat{y} and y , which is needed to use some of the machine learning algorithms. The loss function is in that case defined on the superset of \mathcal{Y} . To obtain a value of prediction from \mathcal{Y} , we apply some projection t to the output \hat{y} . An example of such projection is thresholding, e.g. for $\mathcal{Y} = \{0, 1\}$ and the output of f real valued, the projection function is

$$t(\hat{y}) = \begin{cases} 0 & \text{if } \hat{y} \leq 0.5, \\ 1 & \text{if } \hat{y} > 0.5. \end{cases}$$

Regarding the loss functions, one of the most general lost functions it the is the *0-1 loss*, which for any of its input arguments returns 0 if they match and 1 otherwise. That is

$$\ell^{ZO}(\hat{y}, y) = \begin{cases} 0 & \text{if } \hat{y} = y \\ 1 & \text{if } \hat{y} \neq y. \end{cases}$$

The 0-1 loss function naturally captures the *accuracy* of predictor f – the portion of samples which are predicted correctly by f . Accuracy of the predictor f can be obtained from the 0-1 loss as

$$1 - \mathbb{E}_{(x,y) \sim P^{all}} [\ell^{ZO}(f(x), y)].$$

We want to maximize the accuracy, which is equivalent to minimizing the 0-1 loss.

However, the 0-1 loss has two main drawbacks. First, it does not capture how close is \hat{y} to y – only whether these two matches. Second, the 0-1 loss has zero gradient a.e., which is an obstacle for optimization algorithms computing gradients of the loss function to obtain optimal f , as the value of the gradient is non-informative. In this case, loss functions with more informative gradient values are used instead. An example of such a loss function is the squared error, which is defined as

$$\ell^{SE}(\hat{y}, y) = \|\hat{y} - y\|^2.$$

Note that this loss function continuously decreases as \hat{y} gets closer to y . Therefore, minimizing this loss approximates maximizing the accuracy.

Other frequently used loss functions include cross-entropy or absolute error [8].

A careful reader might wonder why we do not predict y from x exactly. The explanation is that it is often practically impossible or intractable to find such a predictor. First, the exact solution to the problem may not be present in the hypothesis space \mathcal{F} . Second, there may be no practically usable closed-form formula to compute f directly, and \mathcal{F} may be too big to allow for an exhaustive search. The above formulation of the problem allows us to 1. obtain a result even in the case when there is no exact solution to the problem in the set \mathcal{F} , and if needed, 2. use faster methods which usually find only an approximate solution, but do so in a reasonable time.

Now we can summarize the above objective as the *risk minimization objective* [9]:

Definition 2.1 (Risk minimization objective)

The risk minimization objective is to minimize the risk functional

$$R^{all}(f) = \mathbb{E}_{(x,y) \sim P^{all}} [\ell(f(x), y)] \quad (2.1)$$

by the choice of $f \in \mathcal{F}$.

From the practical point of view, this objective has a significant drawback: the distribution P^{all} is often not known [9], or searching for a suitable f over the distribution is not tractable (calculation of the expectation in (2.1) means calculation of an integral, which generally may not have an analytic solution). Thus, we often cannot minimize (2.1) directly.

Therefore a finite *dataset* \mathcal{D} is drawn independently from the distribution. Let $n_d \in \mathbb{N}$ denote the number of samples in \mathcal{D} . Then the *empirical risk minimization* (ERM) *objective* is solved [9]:

Definition 2.2 (Empirical risk minimization objective)

Minimize the empirical risk functional

$$R^{ERM}(f) = \frac{1}{n_d} \sum_{(x,y) \in \mathcal{D}} [\ell(f(x), y)] \quad (2.2)$$

by the choice of $f \in \mathcal{F}$, where \mathcal{D} is a finite dataset drawn i.i.d. from the distribution P^{all} .

The process of selecting the predictor $f \in \mathcal{F}$ is called training.

The rationale behind using \mathcal{D} to represent P^{all} is that \mathcal{D} forms an estimate \hat{P}_{all} of P^{all} . For $n_d \rightarrow \infty$, \hat{P}_{all} converges to P^{all} almost surely. This is a solid reason to think that the low values of the empirical risk minimization

objective (2.2) imply low values of the risk minimization objective (2.1) for large enough datasets. As we will show later in this chapter, some additional assumptions need to be made for the above-mentioned to hold, but we can say in advance that for a wide class of problems, it does.

There are many hypothesis spaces \mathcal{F} used, namely the linear regressors, support vector machines, or neural networks [7]. Each of these is connected with methods used for minimizing the (2.2) term. In the experimental section of this thesis, we will use the linear regression solved by the least-squares method [10] and the feedforward neural network solved by the gradient descent algorithm [7].

Example 2.1 (Linear regression and least squares method). Let $\mathcal{D} \subset \mathcal{X} \times \mathcal{Y}$ be a dataset of size $n_d \in \mathbb{N}$. We will assume that the space of the labels \mathcal{Y} is a subset of \mathbb{R} , that is, the labels are real numbers.

We will arrange the samples from \mathcal{D} into matrices $\mathbb{X} \in \mathbb{R}^{n_d \times n}$, $\mathbb{Y} \in \mathbb{R}^{n_d}$, where the first row of \mathbb{X} is the x element of the first sample $(x, y) \in \mathcal{D}$ and the first element of \mathbb{Y} is the corresponding label y . The second row of \mathbb{X} is the x element of the second sample, and the second element of \mathbb{Y} is the corresponding label of y and so on.

Assume the loss function in the form of the squared error ℓ^{SE} .

We want to find a predictor from the hypothesis space of *linear predictors* \mathcal{F} of functions parametrized by a vector $v \in \mathbb{R}^n$ in the form of $f_v(x) = v^\top x$ for all $x \in \mathbb{R}^n$.

In [10] it is shown that for \mathbb{X} with linearly independent columns

$$w = (\mathbb{X}^\top \mathbb{X})^{-1} \mathbb{X}^\top \mathbb{Y} \in \mathbb{R}^n, \quad (2.3)$$

is the minimizer of the objective function $R^{LSM}(v)$

$$R^{LSM}(v) = \sum_{(x,y) \in \mathcal{D}} \ell^{SE}(v^\top x, y). \quad (2.4)$$

By dividing $R^{LSM}(v)$ by the number of samples n_d , we get the empirical risk minimization objective (2.2) for a loss function ℓ^{SE} and the hypothesis space of linear predictors \mathcal{F} . Because division and multiplication by a positive number do not change the argument of the minima, w is a minimizer of the empirical risk minimization objective with loss function ℓ^{SE} and hypothesis space \mathcal{F} .

In the experimental section of this thesis we will use a modified solution in the form

$$w = (\mathbb{X}^\top \mathbb{X} + \lambda \mathbb{I})^{-1} \mathbb{X}^\top \mathbb{Y} \in \mathbb{R}^n, \quad (2.5)$$

where $\lambda \in \mathbb{R}_0^+$ is a regularisation parameter. Equation (2.5) has two interpretations. First, the computation of the matrix inverse $(\mathbb{X}^\top \mathbb{X})^{-1}$ is unstable in

computer arithmetic especially for ill-conditioned (nearly singular) matrices \mathbb{X} . Adding some positive number to the matrix diagonal makes the matrix diagonally dominant (diagonally dominant matrices are nonsingular). This allows for a more stable computation of the inverse. Second, it can be shown [11] that (2.5) is the minimizer of

$$R^{ridge}(v) = \sum_{(x,y) \in \mathcal{D}} \left(\ell^{SE}(v^\top x, y) \right) + \lambda * \|v\|_2^2. \quad (2.6)$$

We can see that an additional constraint on the value of the regression coefficients was added to the objective function (2.4). λ influences the strength of the constraint – larger λ pushes the coefficients more towards zero. For ill-conditioned problems (e.g., for problems where the number of the parameters of the model is significantly larger than the number of samples available for training), the additional constraint bounds the set of suitable solutions w in contrast to the original problem (2.4) [11].

Example 2.2 (Feedforward neural network and gradient descent [7]). The *feedforward neural network* can be described as a function, which is a composition of its *layers*

$$f_\theta = f_{\theta_p}^{(p)} \circ \dots \circ f_{\theta_2}^{(2)} \circ f_{\theta_1}^{(1)},$$

for some $p \in \mathbb{N}$, where for each $i \in \{1, 2, \dots, p\}$ layer $f_{\theta_i}^{(i)}$ is a function $f_{\theta_i}^{(i)} : \mathbb{R}^{q_{i-1}} \rightarrow \mathbb{R}^{q_i}$ parameterized by a vector of parameters $\theta_i \in \mathbb{R}^{r_i}$, $r_i \in \mathbb{N}$. For all $i \in \{0, 1, \dots, p\}$, $q_i \in \mathbb{N}$. We put $q_0 = n$ and $q_p = m$ (the dimensions of \mathcal{X} and \mathcal{Y} respectively). More elaborate definition of the feedforward neural network and the commonly used layers is beyond the scope of this text. We can refer the reader to [7] or [8].

In the feedforward neural network, the input x goes through each layer without returning to some of the previous layers.

We will assume that each layer is differentiable a.e. in its domain. Let $\theta := \theta_1 \times \theta_2 \times \dots \times \theta_p$. For a differentiable loss function ℓ , we can compute the expected value of the gradient of the loss function over the dataset \mathcal{D} w.r.t the model parameters

$$g(\theta) = \frac{1}{n_d} \sum_{(x,y) \in \mathcal{D}} \nabla_\theta \ell(f_\theta(x), y),$$

which is the gradient of the $R^{ERM}(f_\theta)$ term (2.2) with respect to θ .

By setting $\theta \leftarrow \theta - \epsilon g(\theta)$, where ϵ is called *learning step*, the ERM term decreases. The learning step is repeated until the ERM term reaches sufficiently low value. This method is called *gradient descent*.

Because the computational complexity of the computation of g is $O(n_d)$, usually a randomly selected subset \mathcal{D}' of \mathcal{D} is used instead:

$$g(\theta) = \frac{1}{n'_d} \sum_{(x,y) \in \mathcal{D}'} \nabla_{\theta} \ell(f_{\theta}(x), y),$$

where n'_d is the size of \mathcal{D}' . This reduces the computational complexity of each iteration, and the method is called *stochastic gradient descent*.

Note that the gradient descent is not guaranteed to reach global minima. However, it is often enough to find local minima with a sufficiently low value of R^{ERM} .

Machine learning models based on the ERM theory are largely adopted. This is because the ERM theory puts very loose constraints on the solved problem – for example, there is no constraint on P^{all} – and many problems and methods fit the ERM framework. Furthermore, the ERM-based methods are backed by more than forty years of research and provided by many software frameworks.

2.1 The Theory Behind Empirical Risk Minimization

The general risk minimization objective (2.1) and the empirical risk minimization objective (2.2) were introduced at the beginning of this chapter. Below, we will explain how the latter approximates the former.

First, let us define the notion of the VC (Vapnik-Chervonenkis) dimension and related concepts.

Throughout this section, we assume the predictor with the output in the form of $\mathcal{Y} = \{\pm 1\}$ – this is the *binary classification task*, where we want to match the input data with one of two possible classes. We also assume that each function $f \in \mathcal{F}$ is a mapping $f : \mathcal{X} \rightarrow \mathcal{Y}$. Further, we assume that the loss function in the R^{all} and R^{ERM} terms is the 0-1 loss function ℓ^{ZO} . **We have no assumptions on the form of the distribution P^{all}** , which makes the theoretical results very general.

For the purpose of this section only, *input tuple* $\mathcal{S} = (x_1, \dots, x_{n_s})$, $n_s \in \mathbb{N}$, denotes a tuple of input data x sampled i.i.d. from the marginal probability distribution of the inputs P_X^{all} , where P_X^{all} is obtained from P^{all} by marginalizing out the labels y .

Definition 2.3 (Dichotomies of \mathcal{S} [12])

The set of dichotomies $\Pi_{\mathcal{F}}(\mathcal{S})$ for input tuple \mathcal{S} and hypothesis space \mathcal{F} is defined

$$\Pi_{\mathcal{F}}(\mathcal{S}) := \{(f(x_1), \dots, f(x_{n_s})) | f \in \mathcal{F}\}$$

This represents the set of all possible outputs of functions in \mathcal{F} applied to the elements of \mathcal{S} .

Definition 2.4 ([12])

We will say that the input tuple \mathcal{S} is shattered by \mathcal{F} if and only if $|\Pi_{\mathcal{F}}(\mathcal{S})| = 2^{n_s}$.

This means that for this specific $\mathcal{S} = (x_1, \dots, x_{n_s})$ and for **each possible** tuple of labels $(y_1, \dots, y_{n_s}) \in \mathcal{Y}^{n_s}$ there exists a predictor f in \mathcal{F} , such that $(f(x_1), \dots, f(x_{n_s})) = (y_1, \dots, y_{n_s})$. In other words, for any dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n_s}$, where x_i in the dataset corresponds to x_i in the shattered input tuple \mathcal{S} and labels $y_i \in \{\pm 1\}$ are arbitrary, there exists a predictor in the hypothesis space \mathcal{F} which predicts y_i from x_i exactly.

Definition 2.5 (VC dimension [12])

The VC dimension of the hypothesis space \mathcal{F} is the size of the largest input tuple \mathcal{S} shattered by \mathcal{F} .

If arbitrary size finite input tuples can be shattered by \mathcal{F} then the VC dimension is $+\infty$.

VC dimension can be understood as a measure of the expressive power of the hypothesis space \mathcal{F} . The larger the VC dimension, the larger the dataset size that some function from \mathcal{F} can memorize perfectly. Also, for a hypothesis space with a finite VC dimension d and for any sample $\mathcal{S} = (x_1, \dots, x_{n_s})$ of size n_s larger than d , there exists some labeling of the sample \mathcal{S} (i.e., each sample $x \in \mathcal{S}$ is assigned a label $y \in \mathcal{Y}$), for which there is no function $f \in \mathcal{F}$ with outputs of f matching the labeling on all data from the sample \mathcal{S} .

VC dimension is often proportional to the number of parameters of the hypothesis space \mathcal{F} . Many practically used hypothesis spaces have finite VC dimensions. The VC dimension of the linear predictor $f_v(x) = v^\top x$ with binary output is d for $v \in \mathbb{R}^d$ [8]. The VC dimension of a feedforward neural network with sign activation function and E parameters is $O(E \ln E)$ [8]. In [13], the upper bound on the VC dimension for neural networks with ReLU activations is shown.

Now we introduce the key theorem connecting the (2.1) and (2.2) objectives:

Theorem 2.1 ([12])

Let \mathcal{F} be a hypothesis space of VC-dimension $d < \infty$, and assume that a random dataset \mathcal{D} of size n_d is sampled i.i.d. from P^{all} , where $n_d \geq d \geq 1$. Let $1 > \delta > 0$, $\epsilon > 0$.

Then with probability at least $1 - \delta$

$$R^{all}(f) \leq R^{ERM}(f) + O\left(\sqrt{\frac{d \ln(n_d/d) + \ln(1/\delta)}{n_d}}\right) \quad (2.7)$$

for all $f \in \mathcal{F}$.

O is the Bachmann-Landau notation ¹.

Theorem 2.1 gives an upper bound on the R^{all} risk minimization objective. It determines how well the predictor resulting from the minimization of the (2.2) objective can generalize to data from the distribution P^{all} unseen in the dataset \mathcal{D} .

There are several ways to lower the upper bound in (2.7) on the R^{all} term and improve the generalization properties of f :

- Predictor f , which better fits the data \mathcal{D} , achieves lower R^{ERM} and therefore decreases the right-hand side of (2.7).
- The larger is the dataset \mathcal{D} , the lower is the right-hand side of (2.7).
- Simpler hypotheses spaces with lower VC dimension decrease the right-hand side.

There are several aspects and consequences of the theorem 2.1 which we will discuss below [12].

Note that the bounds in theorem 2.1 hold with a probability $1 - \delta$. Further, the lower δ (and larger the probability of the equation to hold), the larger the upper bound. This is because the dataset \mathcal{D} can be sampled poorly with some non-zero probability, so it does not represent the underlying P^{all} well, and the estimate given by the R^{ERM} term is of poor quality. For example, the dataset can be sampled in such a manner that one type of input is almost missing. The probability of such a situation decreases with the growing size of the dataset.

Furthermore, the theorem assumed i.i.d. sampling of \mathcal{D} , which can be violated. For example, the data collection process can introduce an unwanted bias. Consider the case of handwritten digit recognition. The participants

¹ $h(x) = O(g(x))$ means that there exists a positive real number $C > 0$ and real number N such that $|h(x)| \leq C|g(x)|$ for all $x \geq N$.

who provide the digit samples can try to write neatly. However, this is rarely the case in real life. The predictor trained only on neat handwriting can be unable to recognize the real-life samples, which look different.

Hypothesis spaces with infinite VC dimensions are very likely to fit the dataset \mathcal{D} perfectly without any guarantees on their generalization properties to data sampled from P^{all} not included in \mathcal{D} . This is a phenomenon strongly correlated with *overfitting*. Overfitting is a situation in which the empirical risk R^{ERM} is low, yet the risk R^{all} is high. This is because, with the growing VC dimension of the hypothesis space, a more complex predictor allows to fit the training data better, which leads to lower risk R^{ERM} . Yet, the upper bound on R^{all} is higher, as the more complex predictor allows to memorize the dataset \mathcal{D} without capturing the underlying data generating distribution, achieving low accuracy on the data from P^{all} not included in the dataset \mathcal{D} as a result.

As mentioned earlier, many practically used hypothesis spaces have finite VC dimensions; therefore, they do not violate the theorem’s assumptions.

Also, notice that the equation holds only for datasets larger or equal in size to the VC dimension of the hypothesis space. For datasets smaller than the VC dimension, some predictor from the hypothesis space \mathcal{F} may fit the data perfectly, yet no useful upper bound on R^{all} can be given. This effect also strongly correlates to overfitting – using too rich hypothesis space on a small dataset can produce a predictor memorizing the dataset without any generalization capabilities to unseen data. Note that the datasets in the experimental section of this thesis *are* smaller than the VC dimension of the hypothesis spaces used, and the effect of overfitting is observable: the accuracy on data exposed during training is nearly 100%, yet the accuracy on data not exposed during training is significantly smaller. However, the predictors are still mostly able to provide useful estimates of the labels on data not exposed during training and are therefore useful.

The upper bound (2.7) is overestimated for most practical applications. This is because the upper bound must also work for the worst cases of P^{all} , which may not be very usual in practical applications. In many practical applications, hypothesis spaces with large VC dimensions are used without more significant deterioration of the generalization properties to the data from P^{all} unseen in \mathcal{D} .

Sometimes we can notice that the risk R^{all} is higher than R^{ERM} . This is caused not only by the second term on the right-hand side of (2.7) but also by the fact that the minimization procedure choosing the predictor $f \in \mathcal{F}$ minimizing R^{ERM} favors predictors whose risks R^{ERM} are by chance lower than their risk R^{all} on the specific instance of \mathcal{D} .

It is straightforward to see that the above holds only in the case where we

do not assume the underlying probability distribution P^{all} to change. If the predictor is trained on data from probability distribution A and evaluated on data from arbitrary probability distribution B, the upper bound on the risk does not hold, and the risk can be arbitrary. This is the phenomenon of cover source mismatch in image steganography, where the assumed probability distribution of the cover images used to obtain a predictor for steganalysis is different from the actual probability distribution used by the steganograph. Therefore, we need to utilize further information about the structure of the problem to circumvent this issue, which we do by using the concept of environments as shown in the next chapter.

Chapter 3

Incorporating Environments Into Prediction Tasks

This section will study how the partitioning of the data into environments can be utilized in the prediction tasks. First, we will discuss measures of quality of prediction in the presence of environments. Then we will discuss learning objectives incorporating the knowledge of the environment.

In the experimental part of this thesis (chapter 4), we show how environments can describe the problem of the cover source mismatch, and we demonstrate, evaluate and compare the proposed methods on the problem.

The following notation largely follows [14]. Let \mathcal{E}_{all} denote a set of all *environments*. It can be, for example, the set of all people in the world who can write digits. $e \in \mathcal{E}_{all}$ will denote a single environment (for example, a single participant in the handwritten digits dataset). Let $\mathcal{E}_{tr} \subset \mathcal{E}_{all}$ be the set of all *training environments* – environments directly available for training. This can be a set of all participants who provided their writing samples when the dataset of handwritten digits was created. This dataset is then used to train the predictor for handwritten digit recognition. Specifically, let $\mathcal{D}_e = \{(x_i^e, y_i^e)\}_{i=1}^{n_e}$ be a dataset for an environment e , where n_e denotes the number of samples in the dataset for this environment. We will assume that \mathcal{D}_e are i.i.d. samples drawn from some joint probability distribution P^e over $\mathcal{X} \times \mathcal{Y}$.

Note that \mathcal{E}_{all} can be significantly larger than \mathcal{E}_{tr} . For example, it would be impossible to collect digits samples from all people in the world and store them centrally. There will always be environments that we have never seen during training, similarly to the samples from P^{all} unseen in the dataset \mathcal{D} in the ERM. As we will see in the application domain of image steganography in chapter 4, different environments (called cover sources) are formed by a broad spectrum of parameters, starting with the captured scene, camera

model and camera settings and ending with the software used to post-process images and JPEG quality factor of the resulting image. An accurate model of the images is not available, and the number of combinations of the possible parameters is enormous, which restricts us from using a significant fraction of them for training. Therefore, the set of training environments is significantly reduced compared to the set of all environments.

3.1 Measures of Quality of the Predictor

Measures of the quality of the predictor capture our goals in the specific problem. Their choice also influences the suitability of specific algorithms to obtain predictors achieving good results under these measures. We will assume a general set of environments \mathcal{E} , over which are the values computed.

Risk functional: If the probability distribution of the environments P^{ENV} is known, the probability distribution P^{all} of the data across all of the environments is given by

$$P^{all}(x, y) = \sum_{e \in \mathcal{E}} P^e(x, y) P^{ENV}(e). \quad (3.1)$$

To capture the quality of the predictor, we can then use the risk functional (2.1). The value of the risk functional is not influenced by high loss function values in environments with sufficiently low probability of occurrence.

If the probability distribution P^{ENV} can be estimated or is known, then the risk functional is the best estimate of the performance of the predictor on data from P^{all} . However, the probability distribution P^{ENV} is often not known – and if it were, the problem would reduce to the empirical risk minimization problem and would not be interesting anymore. If a wrong estimate of P^{ENV} is used, the values of the risk functional can be significantly different.

Further, the value of risk functional may not be relevant in cases where the adversary is able to influence the probability distribution P^{ENV} , e.g. by being able to select an environment he uses to his benefit. Let R^e denote *risk in/under/for environment e*:

$$R^e(f) = \mathbb{E}_{(x,y) \sim P^e} [\ell(f(x), y)] \quad (3.2)$$

The adversary can utilize his knowledge of the predictor f to select an environment with maximum $R^e(f)$. In the domain of image steganography, the steganograph may select the cover source to with minimize the accuracy of the steganalyst and avoid detection.

Risk of robust prediction: Another measure of the quality of the predictor, is the *risk of robust prediction*

$$R^{rob}(f) = \max_{e \in \mathcal{E}} R^e(f). \quad (3.3)$$

The risk of robust prediction handles both disadvantages of the risk functional. It is independent of the probability of the environments P^{ENV} , which is useful if the probability of the occurrence of the environments cannot be accurately estimated, as the approach puts an upper bound on the worst-case among these environments. The upper bound on the worst case is also an advantage if the presence of an adversary is assumed. Minimizing the risk of robust prediction minimizes the risk $R^{\tilde{e}}$ for the worst possible choice of $\tilde{e} \in \mathcal{E}$ and the upper bound on other risks has a form: $\forall e \in \mathcal{E}, \max_{e' \in \mathcal{E}} R^{e'}(f) = R^{rob}(f) \geq R^e(f)$.

However, the risk of robust prediction (3.3) does not take into account the intrinsic difficulties of the prediction tasks in the environments. The best achievable risk can be higher in some environments than in others, and as minimizing (3.3) minimizes risk only in the environments forming the maximum, the risk in environments not forming the maximum can be significantly higher than the lowest achievable value in the respective environments. This phenomenon is observable in the experimental section of this thesis 4.3, where the risk of robust prediction is minimized by the *minrisk classifier*.

Regret of robust prediction: To circumvent the above mentioned issue, the *optimistic values of risk in environment e* , r_e , are subtracted from the risks R^e . Value of $R^e(f) - r_e$ is called *regret in environment e* and the *regret of robust prediction* is

$$R^{reg}(f) = \max_{e \in \mathcal{E}} (R^e(f) - r_e). \quad (3.4)$$

The optimistic value of risk in environment captures our belief about the best achievable risk in the respective environment. By minimizing the regret of robust prediction (3.4) we try to achieve the optimistic values of risk in all environments. However, determining r_e can be challenging. The estimation of r_e can be performed by training a specialized predictor in each environment and setting r_e to the risk of the specialized predictor in the respective environment, which is computationally expensive. This is also the case in the experimental section of this thesis. The value of the regret in the environment then qualifies how much we lose by not using the predictor explicitly trained for the respective environment.

3.2 Environments in Supervised Machine Learning Methods

In this section, we present how to use the empirical risk minimization framework from chapter 2 to solve supervised machine learning tasks formulated using environments. Then we show other supervised machine learning methods utilizing the structuring of the problem into environments.

Each of the methods has its advantages and disadvantages, which we will discuss, and the suitability of each method is mainly dependent on the specific solved problem.

3.2.1 Clairvoyant Method

Clairvoyant method minimizes the *risk functional* for an estimate \hat{P}^{all} of P^{all} constructed from the data generating probability distributions P^e in environments \mathcal{E}_{tr} . The method utilizes the framework presented in chapter 2. In order to apply the clairvoyant method, we need an estimate of the probability distribution of the environments P^{ENV} , which is a strong assumption, because P^{ENV} , as already mentioned, is generally not known. Further, the quality of the estimate of P^{ENV} is also significant, as different estimates of P^{ENV} lead to different results – this can be seen in figure 3.1a, where the effect, albeit not strong, is noticeable. If the estimate of P^{ENV} is good and if the resulting \hat{P}^{all} is a good estimate of P^{all} , the clairvoyant method is the best to maximize accuracy on P^{all} .

In order to practically use the clairvoyant method, we need to construct a dataset \mathcal{D} over which is the empirical risk computed. For a problem formulated using environments, we can construct \mathcal{D} from the datasets \mathcal{D}_e for environments in the set of training environments \mathcal{E}_{tr} . The construction of the dataset also includes an assumption on the probability of the environments P^{ENV} (projected in the portion of the samples from \mathcal{D}_e relative to the overall number of samples). Sometimes we can notice that a simple union of the datasets for the training environments $\mathcal{D} = \cup_{e \in \mathcal{E}_{tr}} \mathcal{D}_e$ is used. This forms an implicit assumption on the probability distribution P^{ENV} and the probability of each environment is then determined as the portion of its samples to the total number of samples in the union

$$P^{ENV}(e) = \frac{n_e}{\sum_{e \in \mathcal{E}_{tr}} n_e}.$$

The clairvoyant method does not recognize between the environments and may be less suitable to minimize the risk of robust prediction and regret of

robust prediction as a result. Nevertheless, in many applications, the results achieved using the clairvoyant method are sufficient approximations of the solution of the risk or regret of robust prediction.

Another property of the clairvoyant method is that it minimizes the overall risk with respect to the assumed probability of the environment, weighting the risk in more probable environments as more significant. This can be desirable if the environment \tilde{e} , where the value of $R^{\tilde{e}}(f)$ is the largest of all the environments, is very unlikely to appear.

The clairvoyant method is referred to as holistic linear and holistic MLP in the experimental part of this thesis.

3.2.2 Mixture of Experts, Atomistic Approach

A generalization to the clairvoyant method is to combine multiple expert predictors (*experts*) to solve the classification task.

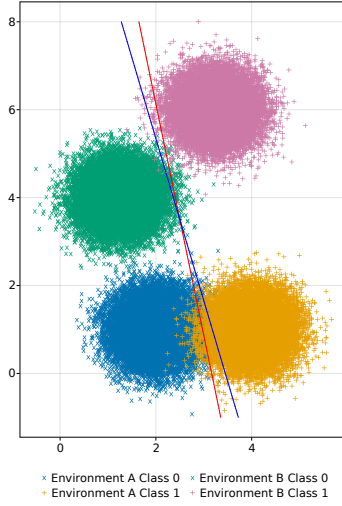
A large part of the prior art literature uses unsupervised learning to divide the data into groups. Therefore, prior knowledge about the environment is not used. Notable is the mixture of experts method [15], which attempts to divide the dataset \mathcal{D} into similar groups and employ an expert predictor on each of these groups.

The *atomistic approach* in image steganography [4] also divides the input domain into multiple parts and applies an expert predictor on each part. We implement the approach in the experimental section of this thesis as a composed predictor, where first, a supervised forensic predictor detects the environment from which the input data originate. The composed predictor's output is then the expert predictor's output for the detected environment. Expert predictor for the environment e is trained only on the dataset \mathcal{D}_e for the environment. We will show that the regret of the resulting predictor is excellent in environments \mathcal{E}_{tr} exposed to the algorithm during training, but regret in environments from $\mathcal{E}_{all} \setminus \mathcal{E}_{tr}$ is significant. Furthermore, the method is computationally expensive, as multiple predictors need to be trained.

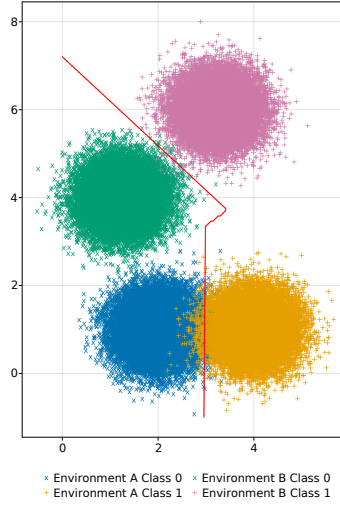
Demonstration of our implementation of the atomistic approach on the data from figure 1.2 is in the figure 3.1b. Both environments were exposed during training, and we can see that the forensic predictor was successful in recognizing the environments and employed suitable predictors in these.

3.2.3 Robust Learning

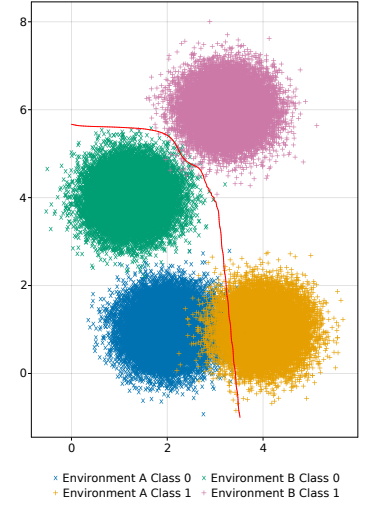
Robust learning objective: The *robust learning* objective is a min max optimization [16] problem:



(a) Clairvoyant method



(b) Atomistic approach



(c) Convex robust learning

Figure 3.1: Comparison of methods from section 3.2. **Figure 3.1a:** Demonstration of the clairvoyant method. Linear predictors obtained from the closed form solution of Ridge regression. Decision boundaries are represented by the red and blue lines. The red line corresponds to the case where the probabilities of the environments are assumed to be the same. The blue line corresponds to the case where the probability of environment A is assumed to be one twelfth of the probability of environment B. The probabilities of the environments were enforced by the train dataset size – train datasets of the same size for both environments in the first case, train dataset of the environment A twelve times smaller than the train dataset of environment B in the second case. **Figure 3.1b:** Demonstration of the atomistic approach. The red line is the decision boundary of the resulting composed predictor. The forensic predictor is neural network with two hidden neurons and tanh activation. Specialised predictors are linear predictors obtained using the closed form solution of the Ridge regression. **Figure 3.1c:** Demonstration of the convex robust learning. Red line is the decision boundary of the resulting multilayer perceptron predictor. The optimistic values of risk was estimated by a linear predictor trained using SGD.

Definition 3.1 (Robust learning objective)

Minimize

$$R^{regtr}(f) = \max_{e \in \mathcal{E}_{tr}} (R^e(f) - r_e), \quad (3.5)$$

by the choice of $f \in \mathcal{F}$.

The robust learning objective minimizes the regret of robust prediction (3.4) in the training environments, and if the values of r_e are set to zero, it minimizes the risk of robust prediction 3.3 in the training environments.

As opposed to the clairvoyant method, we do not need to estimate the probability of the environments P^{ENV} . As already mentioned, estimating the optimistic values of risk r_e can be computationally expensive. Furthermore, the quality of the estimate can significantly alter the result, as shown in the experimental part of this thesis in section 4.3 on the *minregret simple validation* and *minregret extended validation* predictors.

In the experiments in chapter 4 we observed the values of the loss functions to be very noisy during the minimization of the robust learning objective. We think that the problem originates in the fixed step size of the gradient descent algorithm used. Assume that the predictor f_θ is parameterized by a vector of parameters θ . Then the gradient of (3.5) w.r.t. θ is $\frac{\partial}{\partial \theta} R^{\tilde{e}}(f_\theta)$, where \tilde{e} is the arg max of (3.5). The gradient contains information on how to decrease regret only in the environment composing the maximum. Without proper line search, the fixed step size gradient descent may accidentally increase the regret in some other environment above the maximum.

Convex robust learning objective: In the following formulation, the maximum is replaced by a convex combination of the regrets, which we hope to resolve the issue mentioned above, as the computation of gradient uses information from all environments with nonzero coefficients in the convex combination:

Definition 3.2 (Convex robust learning objective)

Assume $q = |\mathcal{E}_{tr}| < +\infty$. Assume that the hypothesis space \mathcal{F} is parameterized by the parameter vector θ . The convex robust learning objective is to find the solution (θ, λ) of

$$\min_{\theta} \max_{\lambda \in \Lambda} \sum_{e \in \mathcal{E}_{tr}} (R^e(f_\theta) - r_e), \quad (3.6)$$

where

$$\Lambda = \left\{ (\lambda_1, \dots, \lambda_q) \left| \forall i \in \{1, \dots, q\}, \lambda_i \geq 0 \wedge \sum_{j=1}^q \lambda_j = 1 \right. \right\}. \quad (3.7)$$

Note that as

$$\max_{e \in \mathcal{E}_{tr}} (R^e(f_\theta) - r_e) = \max_{\lambda \in \Lambda} \sum_{e \in \mathcal{E}_{tr}} (R^e(f_\theta) - r_e), \quad (3.8)$$

we have that

$$\min_{\theta} \max_{e \in \mathcal{E}_{tr}} (R^e(f_\theta) - r_e) = \min_{\theta} \theta \max_{\lambda \in \Lambda} \sum_{e \in \mathcal{E}_{tr}} (R^e(f_\theta) - r_e) \quad (3.9)$$

and the solutions of the robust learning objective and the convex robust learning objective are equivalent.

We can see the solution of the convex robust learning objective for the dataset from figure 1.2 in the figure 3.1c. The objective is also demonstrated in the experimental part of this thesis.

Note that the predictors are trained only on the environments \mathcal{E}_{tr} . Upper bound on the risk in environments $e \in \mathcal{E}_{all} \setminus \mathcal{E}_{tr}$ then includes the cases of

- environments e with probability distributions P^e within the convex hull of the training probability distributions [17] [18], and
- environments e with probability distributions P^e within some measurable distance from the training probability distributions [19].

An interesting property connecting the clairvoyant method and robust learning was shown in [14], proposition 2. The minimizer of the regret of robust prediction is a first-order stationary point of the risk functional for a special choice of P^{ENV} (which is represented by the coefficients λ_e in the paper). Unfortunately, to find the special choice P^{ENV} , one has to minimize the regret of robust prediction first.

3.2.4 Domain Adaptation

Domain adaptation [20] transforms the inputs x in the environments in such a manner that the probability distribution of the transformed inputs is as similar as possible across the environments. This (under further assumptions) allows putting an upper bound on the error¹ in all of the transformed environments for a predictor trained in one of the transformed environments. This property is appealing to us, and it is the reason we decided to study this approach.

Large portion of the domain adaptation literature assumes only two environments: *source* environment and *target* environment. **We will therefore**

¹error is the portion of mispredicted samples

follow the two environment setting throughout this subsection. The scenarios with more than two environments are an extension of this setting, and the basic concepts are valid in these too. The rationale behind the two environment setting is inspired by problems where we are provided with a large enough number of (labeled) samples in the source environment. However, in the target environment, we have input data labeled only partially or not at all. The goal is then to create a predictor which works well on both the source and target environments. In the case of an unlabeled target environment, this is achieved by finding a transformation of the inputs so that the inputs in the two environments are mapped on the same probability distribution (in the best case) and then training a predictor on the data from the source environment, expecting the resulting predictor to work well also in the target environment. This is justified by theorem 3.1 introduced later in this section, which under further assumptions puts an upper bound on the generalization error in the target environment.

Majority of this subsection will follow [20] and [21]. e_s denotes the source environment and e_t the target environment. Further we will assume binary classification problem, e.g. $\mathcal{Y} = \{0, 1\}$. We will assume the hypothesis space \mathcal{F} to be a subset of the set of functions $\{f : \mathcal{X} \rightarrow \{0, 1\}\}$.

P_X^e denotes the marginal probability distribution of the inputs x in environment e obtained from the joint probability distribution P^e . The *labeling function* l^e is defined as [20]

$$l^e(x) = \mathbb{E}_y P^e(y|x). \quad (3.10)$$

Note that the value of l^e lies in $[0, 1]$, as the labels can be non-deterministic. In this case, l^e returns the probability of the label being 1. The labeling function forms an optimal Bayesian predictor in the respective environment.

We define the error of the predictor f in environment e w.r.t. the labeling function l as [20]

$$\epsilon_e(f, l) = \mathbb{E}_{x \sim P_X^e} [|f(x) - l(x)|]. \quad (3.11)$$

Specifically, $\epsilon_S(f) := \epsilon_S(f, l^S)$ and $\epsilon_T(f) := \epsilon_T(f, l^T)$. We can see that the error is a value from $[0, 1]$ and the lower is the value, the better does f approximate l on the input data from the respective probability distribution.

Further, we need some measure of the distance of two probability distributions. This can be captured using the total variation distance [20]:

Definition 3.3 (Total variation distance)

Let P and P' be probability distributions defined on the same sigma algebra \mathcal{B} of measurable subsets of the sample space Ω . Then the total variation distance is defined as

$$d_V(P, P') = \sup_{B \in \mathcal{B}} |P(B) - P'(B)|, \quad (3.12)$$

where $P(B)$ denotes the probability of event B under P .

The key theorem of domain adaptation is then

Theorem 3.1 (Upper bound on target error [20])
For any $f \in \mathcal{F}$,

$$\begin{aligned} \epsilon_T(f) \leq & \epsilon_S(f) + \frac{1}{2}d_V(P_X^S, P_X^T) + \\ & + \min \left\{ \mathbb{E}_{x \sim P_X^S} \left[\left| l^S(x) - l^T(x) \right| \right], \mathbb{E}_{x \sim P_X^T} \left[\left| l^S(x) - l^T(x) \right| \right] \right\} \end{aligned} \quad (3.13)$$

The first term is the error of the predictor in the source environment. The second term corresponds to the discrepancy between the marginal probability distributions of the inputs. The third term corresponds to the discrepancy of the labeling functions.

From the theorem, we can see that with increasing distance between the marginal probability distributions of the input data, the upper bound is increased. The mismatch between the labeling function also increases the upper bound. If the second and third terms are small enough and do not make the upper bound vacuous, we can see that minimizing the error in the source environment reduces the upper bound on the error in the target environment.

The practical application of the theorem mentioned above is to create a **transformation function** of the inputs so that the distance of the transformed marginal probabilities is minimized. This will reduce the second term in the upper bound (3.13). Labeling functions for the transformed inputs can be obtained from (3.10), and under the assumption that the discrepancy between these labeling functions is low (the third term on the right-hand side of (3.13)), this will put a strict upper bound on the error in the target environment. We can then train a predictor on the transformed data from the source environment, with the error in the target environment bounded from above by the error in the source environment and the supposedly low values of the second and third terms in the equation (3.13). As we will see later in example 3.1 in this section, the assumption of similar labeling functions can be easily violated. Using this method on the problem then may yield unusable results, even though the original problem may be perfectly solvable.

An important drawback of theorem 3.1 is that the total variation distance cannot be accurately computed from a finite number of samples. The authors of [22] show that for large enough n and any algorithm using only $o(n^{\frac{2}{3}})$ samples to distinguish between two discrete probability distributions over n elements using total variation distance, there exist two probability distributions with large total variation distance that are indistinguishable by

the algorithm. This means that any sample-based algorithm using variation difference to distinguish two probability distributions with infinite support would need an infinite number of samples.

Therefore, ref. [21] proposes the following formulation of the upper bound:

Theorem 3.2 (Upper bound on target error [21])

For any $f \in \mathcal{F}$,

$$\begin{aligned} \epsilon_T(f) \leq & \epsilon_S(f) + d_{\tilde{\mathcal{F}}}(P_X^S, P_X^T) + \\ & + \min \left\{ \mathbb{E}_{x \sim P_X^S} \left[\left| l^S(x) - l^T(x) \right| \right], \mathbb{E}_{x \sim P_X^T} \left[\left| l^S(x) - l^T(x) \right| \right] \right\}, \end{aligned} \quad (3.14)$$

where

$$\tilde{\mathcal{F}} = \{ \text{sgn}(|f(x) - f'(x)| - a) | f, f' \in \mathcal{F}, 0 \leq a \leq 1 \}. \quad (3.15)$$

Overall, the upper bound is very similar to the original upper bound (3.13), with the three terms having analogous interpretations [21].

We return briefly to the application of domain adaptation to multiple (two and more) environments in the problem of cover source mismatch in image steganography. To apply domain adaptation to the problem of cover source mismatch, we find the projection of the marginal probability distributions of the inputs x of the cover sources from the set of cover sources \mathcal{E}_{tr} available for training, such that the marginal probability distribution of the transformed inputs matches for all cover sources from \mathcal{E}_{tr} . Then we train a predictor over the transformed inputs in one of these cover sources. The theorems presented in this section then allow us to estimate the upper bound on the error of the resulting predictor on each of the cover sources from \mathcal{E}_{tr} . If we encounter a new environment e not present in the set \mathcal{E}_{tr} , we will apply the previously determined projection to the inputs in this environment, and if we determine the data generating probability distribution in this environment, we will be able to obtain an upper bound on the error of the predictor in this environment.

Drawbacks: The mismatch between the labeling functions, captured by the third term of (3.13) and (3.14), can be very problematic in image steganography if the natural steganography (e.g. [23]) is used. Natural steganography tries to create stego objects (objects created from cover objects by embedding a secret message), which are very similar to cover objects from some different cover source than the one from which the stego objects were created. This could render the above inequalities (3.13) and (3.14) useless when applied to such two cover sources, as the labeling of these objects may be opposite in

the environments – if the cover and stego objects are of the same probability, the third term in the inequality becomes at least 0.5, which renders the inequality unusable, as the error of 0.5 corresponds to random guessing of the class.

The problem of high discrepancy between the labeling functions is also shown in the following example:

Example 3.1 (When the domain adaptation fails [21]). Let $\mathcal{X} = \mathbb{R}$, $\mathcal{Y} = \{0, 1\}$. Consider the input data and labels in the following form:

$$\begin{aligned} P_X^S &= U(-1, 0) \\ P_X^T &= U(1, 2) \\ l^S(x) &= \begin{cases} 0, & x \leq -\frac{1}{2} \\ 1, & x > -\frac{1}{2} \end{cases} \\ l^T(x) &= \begin{cases} 0, & x \geq \frac{3}{2} \\ 1, & x < \frac{3}{2} \end{cases} \end{aligned}$$

Function $f^*(x) = 1 \iff x \in \left(-\frac{1}{2}, \frac{3}{2}\right)$ forms a perfect predictor for the problem, achieving zero error in both environments.

Now assume that we try to apply domain adaptation to the problem and try to transform the distributions P_X^S and P_X^T to minimize some probabilistic distance between them. The transformation function

$$t(x) = I[x \leq 0](x + 1) + I[x > 0](x - 1)$$

transforms the inputs so that $t(P_X^S) = t(P_X^T) = U(0, 1)$, and any measure of distance between the distributions is zero. The goal of the domain adaptation was perfectly achieved, yet we obtain for any predictor $f : \mathbb{R} \rightarrow \mathcal{Y}$

$$\epsilon_S(f \circ t) + \epsilon_T(f \circ t) = 1,$$

as the labeling functions in the environments with transformed inputs give opposite labels. The third term in the upper bound (3.14) is 0.5, therefore the upper bound is vacuous and the above results do not contradict it.

We can conclude the example by stating that by using the domain adaptation (and by extension, any method enforcing the agreement of the statistics of the inputs), the previously tractable problem became intractable.

Another significant drawback to the domain adaptation approach is the following. Imagine the case of two environments, A and B. In environment B, the probability density functions of the two classes significantly overlap,

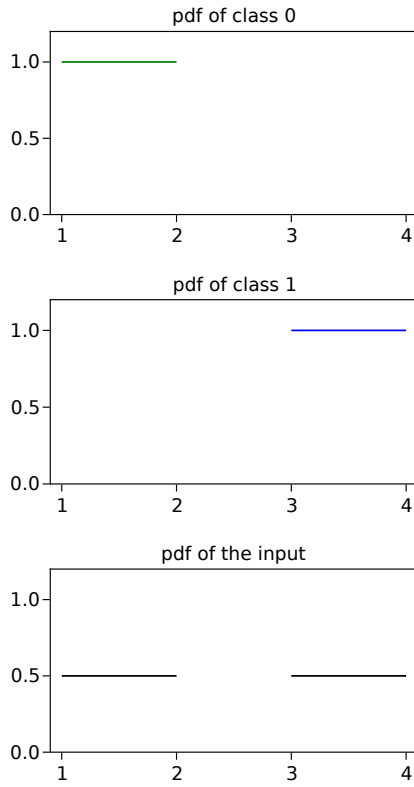
as opposed to environment A, where the overlap is not significant or is not present at all (e.g., the example from figure 3.2). Assume both classes have equal prior probability in both environments. For the inputs in the area of the overlap, the optimal Bayes predictor has no better option than randomly guessing the class of the input. The overlap reduces the predictor’s accuracy, and the larger portion of the samples comes from the overlap, the lower the accuracy.

Assume that the optimal Bayes predictor achieves accuracy α in environment A and β in environment B, $\alpha > \beta$. Now assume that we apply domain adaptation to the problem. We would like to find a projection function, which projects the inputs in environments A and B so that the probability density function of the inputs is identical in both environments after the projection. Such admissible projection is a function, which maps environment A onto environment B. For the example in the figure 3.2, this would be

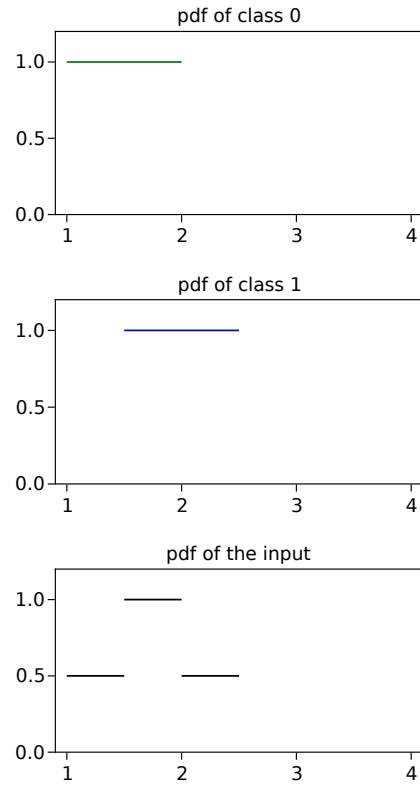
$$t(x) = \begin{cases} x & \text{for } x \in [1, 3], \\ x - 1.5 & \text{for } x \in [3, 4]. \end{cases} \quad (3.16)$$

The optimal Bayes predictor on the transformed data has accuracy β both in environments A and B. As the accuracy of the optimal Bayes predictor is the upper bound on the accuracy of any predictor using the same data to classify the inputs [24], we have that no predictor can achieve better accuracy than β in the transformed environments. Therefore, in environment A, we have regressed from the best achievable accuracy α to β .

Conclusion: We spent plenty of time examining the approach before encountering the drawbacks mentioned above. We think that the low accuracies of the environment-specific predictors in some environments, e.g., for the quality factors 100 in table 4.3 from section 4.3, are caused by the overlap of the probability distributions of the two classes to be recognized. By applying the domain adaptation approach to the problem, accuracy in the environments with good accuracy would then be reduced, as in the example above. This discouraged us from applying the domain adaptation to the problem of cover source mismatch. We see the domain adaptation as an interesting research direction and like its theoretical properties. We hope that further research will be able to modify the theory to be applicable to the problem of cover source mismatch.



(a) Environment A



(b) Environment B

Figure 3.2: Examples of the probability density functions of the inputs in environments A and B. The last row is the probability density function of the inputs if the prior probability of the classes is assumed to be equal.

Chapter 4

Application in Image Steganography

In this chapter, we will expand on the problem of *cover source mismatch* in image steganography¹. The problem is well known within the steganography community, yet very few solutions have been proposed. We believe that the cover source mismatch can be viewed as a sensitivity of the predictor to the mismatch between the training and testing environments. We hope that our experiments will provide original results.

First, let us introduce some basic concepts of steganography. Throughout this chapter, we will call *classifier* the predictor with categorical output.

4.1 Short Introduction to Steganography

The goal of steganography is to communicate a secret message through an overt channel without the secret message being detected [3].

Overview of the whole setting is in the figure 4.1. The sender – called the *steganographer* or simply Alice [25], has an overt *communication channel* and a source of *cover objects*. The cover objects are by themselves benign and include data that the communicating parties do not hesitate to expose to third parties. Examples include common photographs, ordinary written correspondence, or DNS requests. Alice wants to send some *secret message* covertly through the channel. She does so by *embedding* the secret message in one or more cover objects from her communication channel using an *embedding algorithm*. The resulting object is referred to as a *stego object* – it contains the steganographically embedded message.

¹Image steganography is a large interest of the supervisor of this work, which motivated us to use it as our experimental framework.

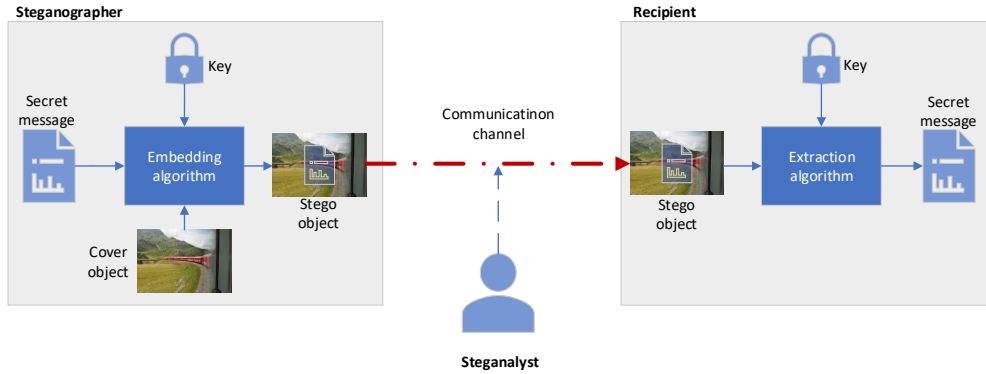


Figure 4.1: Repeated image 1.3. Standard workflow in image steganography.

The overt communication channel is monitored by an adversary, called *steganalyst* or warden. The goal of the steganalyst is to detect whether an embedded secret message *is present* in the object passing through the communication channel *or not*. This is a binary classification task. Notice that the exact contents of the embedded message are not considered in this setting and therefore are of no importance. Then the message can be characterized only by its size and called *payload*.

The *recipient* shares a secret key with the steganographer. The method used to achieve the mutual knowledge of the key is not studied in this setting. The recipient uses his secret key, received stego object, and knowledge of the used embedding algorithm to retrieve the embedded secret message. Following Kerckhoffs's principle [3], the security of the whole communication is expected to rely only on the key.

We assume the contents of the channel to be perfectly observable by all parties, writable by the steganographer, and read-only by the steganalyst [25].

Example 4.1. One of the most simple examples of steganography is the use of invisible ink. In this case, the steganographer's communication channel is an ordinary written correspondence. The cover object is a letter describing her recent vacation. The secret message containing the names and addresses of the members of an espionage network is written in invisible ink on the other side of the paper (the embedding algorithm). In this case, the secret key is not used. Both the steganographer and recipient share the knowledge of the embedding algorithm and know that heating the letter above a candle will make the ink visible (the extraction algorithm). The letter is sent through the postal service. A team of steganalysts is present at the post office and

checks all passing letters. Their goal is to detect whether a letter contains hidden messages or not. A brute force detection algorithm would be to heat all the letters in a furnace and then check whether some secret message has appeared or not.

Example 4.2. All experiments in this work were conducted in a subfield of steganography called *image steganography*. This is because there has been a lot of conducted research in this subfield compared to other subfields of steganography, and therefore there is more literature available.

Image steganography embeds the secret message into images from digital cameras. The cover source is defined by the model of the digital camera with specific settings, the specific scene pictured, parameters of postprocessing and software used, etc. Such fine graining is impractical, and cover sources are therefore mixed to create new cover sources – e.g., a specific digital camera model with a specific ISO sensitivity setting and specific output JPEG quality factor, specific settings of one selected postprocessing software, and other parameters arbitrary.

Message is then embedded using one of many developed embedding algorithms, for example nsF5 [26], J-UNIWARD [27], WOW [28] or HUGO [29].

There are also many well-established sets of steganographic features, for example, cc-JRM [30] and DCTR [31] features, which are then used for steganalysis using the feature-based classification of the images.

4.1.1 Steganalysis and Cover Source Mismatch

As mentioned earlier, the goal of steganalyst is to detect whether an embedded secret message is present in the communication or not. He can do so by constructing a binary classifier for the overt communication channel, which would detect the secret message.

In the literature, it is very often assumed that the steganalyst knows everything about the communication (the steganographic algorithm, message length, and the distribution of cover objects) except the secret key, following Kerckhoffs’s principle [3]. This assumption allows us to develop secure steganographic algorithms.

Unfortunately, the same assumption is commonly made during the evaluation of steganalytic algorithms. In real-world scenarios, the steganalyst cannot accurately estimate the parameters of the used cover source. As a result, this leads to the steganalyst’s assumed cover source being different from the actual cover source used. This situation is called *cover source mismatch* [2], and by using a classifier trained on a different cover source than the one

it is then employed on, the performance of the classifier can be significantly reduced. We try to solve the complications caused by the absence of knowledge about the cover source and mitigate the effect of cover source mismatch by utilizing the concepts presented in previous chapters.

One of the first mentions of the cover source mismatch is found in [32], and yet to date, there is very little progress on the subject. To this day, there are very few publications on this phenomenon. Examples include [2], [5], [33] or [4].

We assume the problem to be caused by the difference between the probability distribution of the cover objects among different cover sources. Our assumption is supported by [2], who have shown that there is a strong statistical footprint of the image development pipeline on the DCT coefficients of the resulting JPEG image. The image processing pipeline is one of the cover source parameters in image steganography. The difference in the statistical footprints supports our hypothesis that the probability distributions of the cover objects in different cover sources are different.

In image steganography, the noise of the resulting image is, for example, influenced by the specific cover source. The camera models differ in the used sensor – its size (smaller the pixels, higher the noise), operating temperature (higher temperature causes more noise), and technology. A higher ISO sensitivity setting uses higher amplification rates of the input signal, which also amplifies the noise. Higher quality factors leave more noise in the image. The JPEG quality factors also differ in the values of the quantization tables used to convert the RAW images to JPEG, which causes differences in the resulting noise.

So far, some of the approaches to mitigating the cover source mismatch were [2]

- *The atomistic approach* explained in section 3.2.2 of this thesis.
- *The holistic approach* trying to construct a single monolithic classifier for the whole problem independent of the specific image processing pipeline. This was done by merging the datasets and training a classifier using a clairvoyant method on the merged data.

However, none of the proposed approaches fully solved the cover source mismatch. We believe that the cover source mismatch problem can be formulated in terms of the environments, where each environment corresponds to one cover source. We can then utilize the methods mentioned in the previous chapter on the problem, which we will do in the second part of this chapter.

4.2 Experimental Details

The problem of cover source mismatch has remained unsolved for more than ten years. We believe that the methods presented in this thesis can solve the problem of cover source mismatch, and we apply these methods to the problem in this section. We will also compare the methods on this problem and evaluate their properties.

4.2.1 Data

We study the following parameters influencing the cover source: specific model of camera, ISO sensitivity setting, and quality factor, e.g., Canon EOS 500D with ISO sensitivity 1600 and quality factor 75. The set \mathcal{E}_{all} is formed by all possible combinations of manufactured digital cameras, their adjustable ISO sensitivity settings, and all possible JPEG quality factors.

The original images were taken from the ALASKA dataset [34]. The images were developed using the following pipeline:

1. The images were converted from raw to TIFF using RawTherapee version 5.4 with the default neutral profile.
2. The TIFF images were cropped to 512×512 pixels.
3. The cropped TIFF images were converted to JPEG with a specified quality factor (value is always specified in the experimental section of this thesis) with 4:4:4 subsampling. The conversion was performed in Python 3.6.9 using Pillow 8.1.2 library.

7 different cameras and ISO sensitivity combinations were selected. They are listed in table 4.1. For better readability, the combinations of the camera and ISO sensitivity in the results were replaced by letters, which are in the respective rows of table 4.1. **The experimental setting utilizes feature-based classification.**

The embedding simulation and feature extraction were performed with the following setup:

- The message length was 0.04 bpAC^2 .
- The embedding algorithm was nsF5 [26].

²bpAC (bits per AC coefficient) is a unit that defines the length of the message in bits relative to the image size – more specifically to the number of AC coefficients of the JPEG image, which is constant for images JPEG images of the same size

| Camera name | ISO | Letter denomination |
|-------------------------|------|------------------------|
| Panasonic FZ28 | 100 | A |
| Nikon D610 | 100 | B |
| iPad Pro (12.9" gen. 2) | 20 | C |
| Canon EOS 500D | 1600 | D |
| Sony ILCE-7R | 800 | E |
| Pentax K10D | 400 | F |
| Panasonic GM1 | 3200 | G |

Table 4.1: Cover sources used for the experiments and their letter denomination in the experimental results.

- The steganographic features were cc-JRM [30].

The nsF5 embedding algorithm simulates optimal coding [35], which can reduce the number of changes made in the image to embed the secret message. For 0.04 bpAC and 512×512 image, the number of changes without optimal coding is approximately 10322. With the optimal coding simulation, the average number of changes for the cover sources we selected for our experiments shrinks to 1163 for quality factor 100 and 2176 for quality factor 75.

The datasets were partitioned into disjoint training, validation, and test datasets in a 3:1:1 ratio. Three different partitionings of the datasets were used, and all reported results are aggregated over these partitionings.

The classifier performance in this section is usually measured as a *probability of error* P_E . This is defined as the ratio of misclassified samples on a dataset made of a 1:1 mixture of cover and stego objects and is a common measure in the steganographic literature – as the exact probability of the stego objects is often not known, equal probability of the cover and stego objects is assumed.

4.2.2 Methods

Three hypothesis spaces were used:

1. *Linear predictor with bias*. The optimization procedure was, in this case, always the same. The inputs were first normalized so that the features had zero mean. The labels were $\{-1, 1\}$. Then the closed-form solution of the Ridge regression with least-squares loss function (see example 2.1, equation (2.5)) for the problem without bias was found. The

bias was then computed using the ROC curve to minimize P_E on the train dataset. The optimal value of the regularization coefficient $\lambda \in \{10^{-7}, 10^{-6}, \dots, 10^{-1}, 1, 10, \dots, 10^4\}$ was selected to minimize P_E on the validation dataset. Because the number of available samples in the datasets we use is much lower than the number of features, making the problem ill-conditioned, we prefer ridge regression to other linear methods for the reasons mentioned in the example 2.1 (better numerical stability, bounded set of solutions).

2. *Multilayer perceptron.* The architecture was selected using a random search on the validation dataset from the set of parameters in the first part of the table 4.2. The random search criterion was always objective dependent and will be specified later in this section, together with the objectives. The optimizer was ADAM optimizer with weight decay. The parameters of the optimizer were selected from the set of parameters in the second part of the table 4.2. Parameters not mentioned were set to their default values.
3. *Multinomial logistic regression.* This served to obtain the optimistic value of risk r_e for the convex robust learning objective, where the regret was computed on loss values, and we needed to compute the optimistic value of risk for the cross-entropy loss. The models were optimised using the ADAM optimiser with default parameters, cross-entropy loss function and L_2 regularization on the weight parameters with parameter $\lambda \in \{10^{-7}, 10^{-6}, \dots, 10^{-1}, 1, 10, \dots, 10^4\}$. The optimal value of λ was selected to minimize P_E on the validation dataset. The number of steps was 200000 with no early stopping.

The experiments were the following:

1. We trained a specialized classifier for each camera A-G and QF 75, 76, 90, and 100 using the linear predictor with a bias to recognize cover and stego images from the respective cover sources. These will be called *single cover source classifiers*. The P_E of the single cover source classifier on the data from the cover source it was trained on is the optimistic value of P_E for the respective cover source.

Further, we implemented several methods from section 3.2 on the problem of recognizing cover and stego images. The *training cover sources* included cameras A-E and quality factors 75 and 100. Other cameras and quality factors were used only to evaluate the resulting classifiers' generalization properties to cameras and quality factors unseen during training and were not

| Model hyperparameters | |
|-------------------------------------|---|
| parameter name | possible values |
| activation function | ReLU, hyperbolic tangent |
| no. of neurons in each hidden layer | 8, 16, 32, 64, 128 |
| no. of hidden layers | 1, 2, 3 |
| Optimizer hyperparameters | |
| parameter name | possible values |
| learning step | $\{2 \times 10^i, 4 \times 10^i, 8 \times 10^i, 10 \times 10^i i \in \{-4, -3, \dots, -6\}\}$ |
| decay parameter | $\{0, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$ |

Table 4.2: Parameter sets for random search of model architecture and optimizer parameters

included in the training and validation sets. The methods utilizing knowledge from multiple cover sources are:

2. We trained three multilayer perceptron classifiers to recognize the cover sources for two versions of the atomistic approach:
 - (a) If the quality factor is known, we need to estimate which camera A-E was used within the QF. We need a classifier for QF 75 and QF 100, a total of 2 classifiers. In the tables, this is *atomistic QF known*. The quality factor can often be read from the metadata of the image, which is why we find this setting interesting.
 - (b) If the quality factor is not known, we need to estimate it with the camera A-E. We need one classifier for it. In the tables, this is *atomistic QF predicted*.

In both cases, the training and validation dataset was constructed as a union of the respective train and validation datasets in the cover sources to be recognized. Early stopping and selection of the best architecture in the random search were selected to maximize the classification accuracy of the validation data. After estimating the camera (and quality factor if needed), we used the corresponding single cover source classifier to determine whether the image is a cover image or a stego image.

3. Two classifiers were obtained using the clairvoyant method. The test and validation datasets were the union of the datasets for the training environments. Note that this includes an implicit assumption on the probability

of the cover sources, where cover sources with larger datasets have a larger probability, as mentioned in the section 3.2.1. These represent the holistic approaches in the literature on steganalysis.

- (a) Clairvoyant method where the hypothesis space is the linear prediction with bias. In the tables, this is *holistic linear*.
 - (b) Clairvoyant method where the hypothesis space is the multilayer perceptron. The early stopping criterion and criterion for selecting the best model was the value of P_E on the validation dataset. In the tables, this is *holistic MLP*. We use the multilayer perceptron because of its non-linearity. We think that the bad performance of the holistic classifiers shown in [2] was caused by the fact that the authors used linear classifiers on a nonlinear problem. Therefore, we want to compare the linear holistic classifier with a nonlinear one.
4. Minimization of the risk of robust prediction in the training cover sources. Multilayer perceptron was used. The risk was estimated on the validation dataset. Early stopping and random search selected the model with the lowest maximum P_E among the validation datasets for the training cover sources. In the tables, this is *minrisk of RP*
 5. Three approaches to minimize the regret of robust prediction. We always used the multilayer perceptron and random search selected classifier with the lowest maximum *risk* among the validation datasets of the training cover sources. The early stopping criterion selected the predictor with the lowest regret value across the training environments. The difference between the approaches was in the estimate of the optimistic value of risk:
 - (a) The *simple train optimistic value of risk*: this is the P_E on the train dataset for the cover source of the single cover source classifier for the respective cover source. The regret is computed as the regret of P_E on the train dataset. In the tables, this is the *minregret simple train*

The P_E of the single cover source classifiers is often very low on the training datasets due to the small number of samples compared to the number of features. We were afraid that the low values of P_E would not allow for a reliable estimate of the environment with maximum regret, and the approach would degenerate to the minimization of the risk. Therefore we also tried the following two estimates of regrets:

- (b) The *simple validation optimistic value of risk* is similar to the simple train optimistic value of risk. However, the optimistic value of risk

| | | A QF75 | G QF75 | A QF100 | G QF100 |
|---|-------|--------|--------|---------|---------|
| A | QF75 | 7.9 | 48.1 | 49.1 | 50.0 |
| G | QF75 | 26.8 | 11.5 | 49.4 | 50.0 |
| A | QF100 | 49.8 | 50.0 | 40.7 | 48.6 |
| G | QF100 | 50.0 | 50.0 | 46.0 | 44.4 |

Table 4.3: Selected results from table A.1. Single cover source classifiers, P_E in percents on the test dataset. Horizontally are cover sources on which the classifiers were trained, vertically are cover sources on which the classifiers were tested.

and regret are computed on the train dataset. In the tables, this is the *minregret simple validation*.

- (c) The *extended validation optimistic value of risk* is estimated as the minimum value of P_E on the validation dataset of the cover source among all the single cover source classifiers for cameras A-E and QF 75 and 100. In the tables, this is the *minregret extended validation*.
6. The minimization of the convex robust learning objective for the train cover sources. In this case, the regrets were computed on the loss functions. The single cover source classifiers were trained on a different loss function and in a manner that did not allow us to obtain a reliable estimate of the optimistic value of risk for the loss function used with the convex robust learning objective. Therefore the optimistic values of risk were estimated from the multinomial logistic regression classifiers, each of which was trained on a single cover source from the set of training cover sources and the optimistic value of risk was calculated on the same cover source. Therefore, they are counterparts of the single cover source classifiers trained using different loss functions.

We used the multilayer perceptron to minimize the convex robust learning objective. Early stopping and random search selected the classifier with the best maximum regret on the validation data. The minimization of the convex robust learning objective is referred to as *minregret convex* in the tables.

4.3 Experimental Results

In the table 4.3 we can notice a strong effect of cover source mismatch. When we test the classifier trained on cover source G QF75 on different

| | camera | quality factor | holistic linear | holistic MLP | minregret simple train | minregret simple validation | minregret extended validation | minrisk of RP | minregret convex | atomistic QF predicted | atomistic QF known |
|-------------|--------|----------------|-----------------|--------------|------------------------|-----------------------------|-------------------------------|---------------|------------------|------------------------|--------------------|
| max regret | A-E | 75, 100 | 11.3 | 7.0 | 2.9 | 1.4 | 2.0 | 24.8 | 1.9 | 0.5 | 0.5 |
| mean P_E | | | 21.4 | 20.4 | 21.0 | 21.3 | 21.3 | 31.7 | 20.2 | 21.0 | 21.0 |
| max P_E | | | 39.5 | 38.9 | 38.7 | 40.4 | 40.6 | 39.5 | 39.6 | 41.6 | 41.6 |
| max regret | A-G | 75, 100 | 11.3 | 7.0 | 4.2 | 3.4 | 2.0 | 35.7 | 1.9 | 11.1 | 9.9 |
| mean P_E | | | 22.3 | 21.6 | 22.4 | 22.7 | 22.4 | 33.6 | 21.5 | 23.2 | 23.2 |
| max P_E | | | 43.8 | 41.7 | 41.9 | 43.3 | 42.3 | 47.2 | 42.7 | 45.2 | 45.0 |
| max regret | A-G | 76 | 13.1 | 21.8 | 12.0 | 22.6 | 15.5 | 28.4 | 21.4 | 28.0 | N/A |
| mean regret | | | 4.5 | 10.3 | 3.9 | 9.3 | 7.0 | 19.7 | 10.1 | 10.3 | N/A |
| max regret | A-G | 77 | 4.1 | 15.1 | 9.4 | 11.7 | 6.7 | 41.8 | 11.9 | 23.8 | N/A |
| mean regret | | | 0.8 | 2.3 | 3.3 | 3.4 | 2.2 | 24.1 | 2.1 | 6.1 | N/A |
| max regret | A-G | 90 | 14.5 | 30.0 | 19.6 | 22.8 | 20.2 | 35.2 | 21.8 | 39.4 | N/A |
| mean regret | | | 7.8 | 14.1 | 9.1 | 10.2 | 9.3 | 27.0 | 11.2 | 21.9 | N/A |

Table 4.4: Aggregated results from tables A.2 and A.3. The column heading for columns 3 to 11 describes the evaluated classifier. All values are measured on the test dataset. Probability of error P_E was measured in percents and the mean and maximum over the cover sources described by the first and second columns is reported. For each cover source, the regret is measured w.r.t. the P_E of the single cover source classifier for the respective cover source it was trained on – positive value means larger P_E than the P_E of the single cover source classifier – and only the maximum and/or mean over the cover sources described by the first and second columns is reported in the table. The regret is reported in percent points. The classifiers from the column headings were trained only on cover sources A to E with QF 75 and 100. Cover sources F and G for any QF and cover sources A-G for QF 76, 77 and 90 were not exposed during training and were used only for test purposes.

cover sources in the table, the P_E steps from 11.5 percent to values near 50 percent, equivalent to random guessing of the class.

The results in the table 4.3 also show that steganography in images with a high quality factor is much less detectable than in images with a low quality factor: P_E of 7.9 and 11.5 for QF 75 vs. 40.7 and 44.4 for QF 100 on the diagonal of the table – the diagonal corresponds to the case when the classifier is tested on the same cover source it was trained on. This is because a higher quality factor leaves more noise in the image, which allows for a more efficient embedding of the message, and as we mentioned earlier, a smaller number of changes is made in the image.

Selected results from the comparison of the methods utilizing knowledge from multiple environments are in table 4.4. We were surprised by negative

values in the table A.2, and we explain them by the fact that the classifiers were provided information from multiple cover sources during training, as these can be more informative than the individual cover source-specific datasets.

The *regret* is the difference between the P_E of the single cover source classifier for the cover source and P_E achieved by the classifier from the respective column of the table 4.4. Maximum regret is reported in the table.

The *atomistic classifiers* are excellent on known cover sources. However, the generalization to cameras and quality factors not exposed during training is very poor.

The *clairvoyant multilayer perceptron* is not significantly better than the *holistic linear classifier*, and on quality factors not exposed during training, it is significantly worse. Therefore we conclude that using nonlinear hypothesis space did not provide a significant advantage.

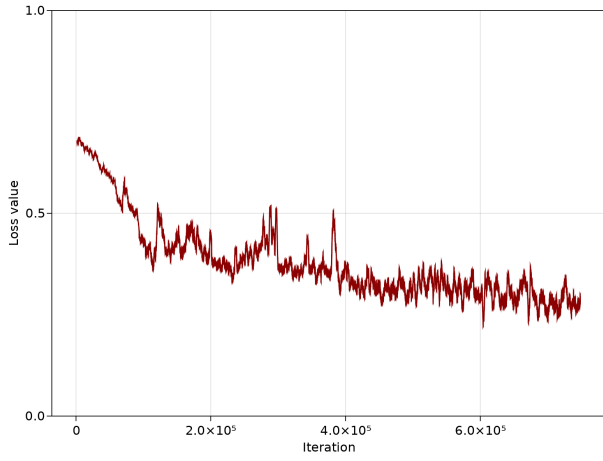
From the minregret classifiers (excluding the minregret convex), the *minregret extended validation* classifier seems to be on par with the *minregret simple train*. As expected, the *minregret **extended** validation* is better than the *minregret **simple** validation*, probably due to the better estimate of the optimistic value of risk, as mentioned in section 3.2.3. The good performance of the *minregret simple train* was not expected, and it seems that the concern about the quality of the estimate of the optimistic value of risk was unfounded.

The *minrisk classifier* reduces the maximum risk in cover sources exposed during training. However, the regret is very large. Both were expected from the analysis in section 3.2.3. It seems that the approach tried to minimize its risk under several cover sources, which were difficult to train by default, while neglecting other cover sources, where it could achieve much lower values of risk.

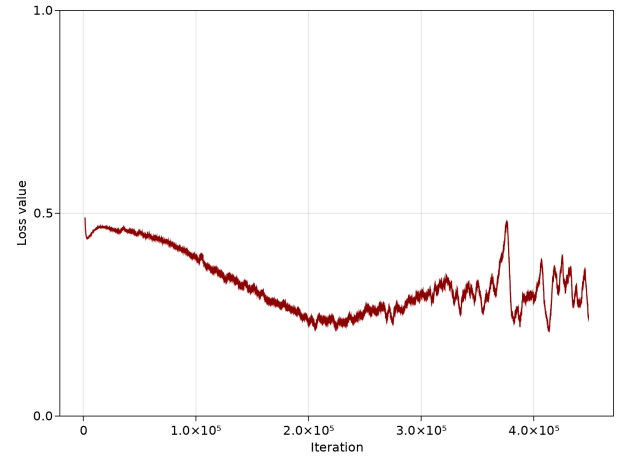
The *minregret convex classifier* generalizes well to cameras not exposed during training on quality factors included in the train dataset both in terms of the maximum regret and mean P_E . This can be explained by 1. better numerical properties of the optimization problem and 2. the fact that the robust learning objective takes into account additional information about the achievable values of P_E in the form of the optimistic values of risk r_e and tries to achieve these.

However, the generalization to quality factors not exposed during training is still best with the holistic linear classifier.

The convex robust learning objective does not remove the noisiness of the values of the loss function (figure 4.3). When comparing smoothed values of the loss functions of the classifiers optimized using the convex robust learning and robust learning objectives (figure 4.2), the convex robust learning objec-



(a) Minregret simple validation



(b) Minregret convex

Figure 4.2: Loss values of the minregret simple validation and minregret convex classifiers on the train dataset in the first dataset partitioning. The values were smoothed by moving average with window size 3001.

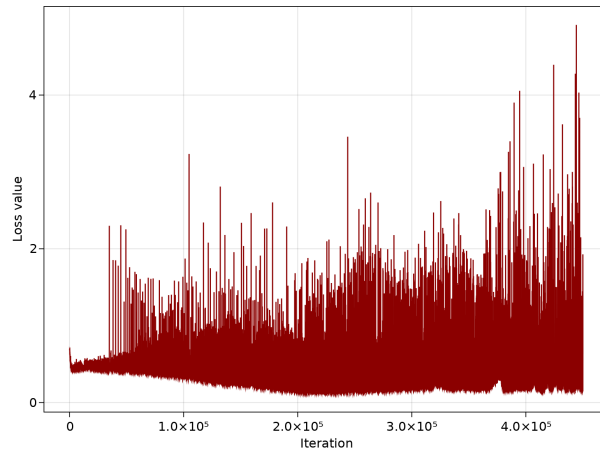


Figure 4.3: Unsmoothed loss values of the minregret convex classifier on the train dataset in the first dataset partitioning.

tive seems slightly smoother, but notice that the smoothing is very strong. The benefit of solving the convex robust learning objective instead of the robust learning objective is not as significant as expected.

Chapter 5

Conclusion

This thesis introduced the reader to basic concepts of supervised machine learning. We discussed methods utilizing structured data in the form of environments to obtain predictors robust to the change in the underlying probability distribution of the data.

After an initial search, we chose the problem of cover source mismatch in image steganography to demonstrate the methods. The problem of cover source mismatch is important and easy to formulate, yet not much attention is devoted to it in the literature. In the experimental section of this thesis, we explained the basic concepts of image steganography. We tested the performance of the methods from previous chapters on the problem of cover source mismatch.

Preparation of the data and implementation of the experimental framework were time-consuming.

Unfortunately, we did not find a truly robust predictor to the change of the cover source, as the results on the quality factors 76, 77, and 90 show. However, the trained predictors were robust to some types of cover source changes. *These results are better than the state-of-the-art literature.* Some of the results were submitted as a conference paper [36] and are under review at the time of writing.

Further Acknowledgments

The software experiments in this work were mostly written in Julia [37] using the CUDA.jl [38] [39], Flux.jl [40] [41] and JuMP [42] packages. The graphs were made using Makie [43]. Special thanks go to Steven G. Johnson for his help with tackling the problem of obtaining the DCT coefficients of a JPEG image in Julia.

Bibliography

1. HEINZI. *Why is 1 hand-written without a serif and 7 without a dash?* 2012. <https://english.stackexchange.com/questions/62586/why-is-1-hand-written-without-a-serif-and-7-without-a-dash>.
2. GIBOULOT, Quentin; COGRANNE, Rémi; BORGHYS, Dirk; BAS, Patrick. Effects and solutions of cover-source mismatch in image steganalysis. *Signal Processing: Image Communication*. 2020, vol. 86, p. 115888.
3. TANENBAUM, Andrew S; BOS, Herbert. *Modern operating systems*. Pearson, 2015.
4. BARNI, M.; CANCELLI, G.; ESPOSITO, A. Forensics aided steganalysis of heterogeneous images. In: *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2010, pp. 1690–1693. Available from DOI: 10.1109/ICASSP.2010.5495494.
5. KER, Andrew D; PEVNÝ, Tomáš. A mishmash of methods for mitigating the model mismatch mess. In: *Media Watermarking, Security, and Forensics 2014*. International Society for Optics and Photonics, 2014, vol. 9028, p. 90280I.
6. LI, Xiaofeng; KONG, Xiangwei; WANG, Bo; GUO, Yanqing; YOU, Xingang. Generalized transfer component analysis for mismatched JPEG steganalysis. In: *2013 IEEE International Conference on Image Processing*. IEEE, 2013, pp. 4432–4436.
7. GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. *Deep learning*. MIT press, 2016.
8. SHALEV-SHWARTZ, Shai; BEN-DAVID, Shai. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.
9. VAPNIK, Vladimir. Principles of risk minimization for learning theory. In: *Advances in neural information processing systems*. 1992, pp. 831–838.

10. FLOUDAS, Christodoulos A; PARDALOS, Panos M. *Encyclopedia of optimization*. Springer Science & Business Media, 2008.
11. HASTIE, Trevor; TIBSHIRANI, Robert; FRIEDMAN, Jerome H; FRIEDMAN, Jerome H. *The elements of statistical learning: data mining, inference, and prediction*. Vol. 2. Springer, 2009.
12. SCHAPIRE, Robert E; FREUND, Yoav. Boosting: Foundations and algorithms. *Kybernetes*. 2013.
13. BARTLETT, Peter L; HARVEY, Nick; LIAW, Christopher; MEHRABIAN, Abbas. Nearly-tight VC-dimension and pseudodimension bounds for piecewise linear neural networks. *The Journal of Machine Learning Research*. 2019, vol. 20, no. 1, pp. 2285–2301.
14. ARJOVSKY, Martin; BOTTOU, Léon; GULRAJANI, Ishaan; LOPEZ-PAZ, David. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*. 2019.
15. MASOUDNIA, Saeed; EBRAHIMPOUR, Reza. Mixture of experts: a literature survey. *Artificial Intelligence Review*. 2014, vol. 42, no. 2, pp. 275–293.
16. LU, Baiquan; CAO, Yuan; ZHOU, Jianzhen, et al. Reference variable methods of solving min–max optimization problems. *Journal of Global Optimization*. 2008, vol. 42, no. 1, pp. 1–21.
17. MOHRI, Mehryar; SIVEK, Gary; SURESH, Ananda Theertha. Agnostic federated learning. In: *International Conference on Machine Learning*. PMLR, 2019, pp. 4615–4625.
18. HOFFMAN, Judy; MOHRI, Mehryar; ZHANG, Ningshan. Algorithms and theory for multiple-source adaptation. *arXiv preprint arXiv:1805.08727*. 2018.
19. STAIB, Matthew; JEGELKA, Stefanie. Distributionally robust optimization and generalization in kernel methods. *Advances in Neural Information Processing Systems*. 2019, vol. 32, pp. 9134–9144.
20. BEN-DAVID, Shai; BLITZER, John; CRAMMER, Koby; KULESZA, Alex; PEREIRA, Fernando; VAUGHAN, Jennifer Wortman. A theory of learning from different domains. *Machine learning*. 2010, vol. 79, no. 1, pp. 151–175.
21. ZHAO, Han; DES COMBES, Remi Tachet; ZHANG, Kun; GORDON, Geoffrey. On learning invariant representations for domain adaptation. In: *International Conference on Machine Learning*. PMLR, 2019, pp. 7523–7532.

22. BATU, Tugkan; FORTNOW, Lance; RUBINFELD, Ronitt; SMITH, Warren D; WHITE, Patrick. Testing that distributions are close. In: *Proceedings 41st Annual Symposium on Foundations of Computer Science*. IEEE, 2000, pp. 259–269.
23. TABURET, Théo; BAS, Patrick; SAWAYA, Wadih; FRIDRICH, Jessica. Natural steganography in JPEG domain with a linear development pipeline. *IEEE Transactions on Information Forensics and Security*. 2020, vol. 16, pp. 173–186.
24. RENGGLI, Cedric; RIMANIC, Luka; HOLLENSTEIN, Nora; ZHANG, Ce. Evaluating Bayes Error Estimators on Read-World Datasets with FeeBee. *arXiv preprint arXiv:2108.13034*. 2021.
25. KER, Andrew D; BAS, Patrick; BÖHME, Rainer; COGRANNE, Rémi; CRAVER, Scott; FILLER, Tomáš; FRIDRICH, Jessica; PEVNÝ, Tomáš. Moving steganography and steganalysis from the laboratory into the real world. In: *Proceedings of the first ACM workshop on Information hiding and multimedia security*. 2013, pp. 45–58.
26. FRIDRICH, Jessica; PEVNÝ, Tomáš; KODOVSKÝ, Jan. Statistically undetectable jpeg steganography: dead ends challenges, and opportunities. In: *Proceedings of the 9th workshop on Multimedia & security*. 2007, pp. 3–14.
27. HOLUB, Vojtěch; FRIDRICH, Jessica; DENEMARK, Tomáš. Universal distortion function for steganography in an arbitrary domain. *EURASIP Journal on Information Security*. 2014, vol. 2014, no. 1, pp. 1–13.
28. HOLUB, Vojtěch; FRIDRICH, Jessica. Designing steganographic distortion using directional filters. In: *2012 IEEE International workshop on information forensics and security (WIFS)*. IEEE, 2012, pp. 234–239.
29. PEVNÝ, Tomáš; FILLER, Tomáš; BAS, Patrick. Using high-dimensional image models to perform highly undetectable steganography. In: *International Workshop on Information Hiding*. Springer, 2010, pp. 161–177.
30. KODOVSKÝ, Jan; FRIDRICH, Jessica. Steganalysis of JPEG images using rich models. In: *Media Watermarking, Security, and Forensics 2012*. International Society for Optics and Photonics, 2012, vol. 8303, 83030A.
31. HOLUB, Vojtěch; FRIDRICH, Jessica. Low-complexity features for JPEG steganalysis using undecimated DCT. *IEEE Transactions on Information Forensics and Security*. 2014, vol. 10, no. 2, pp. 219–228.

32. CANCELLI, Giacomo; DOERR, Gwenael; BARNI, Mauro; COX, Ingemar J. A comparative study of \pm steganalyzers. In: *2008 IEEE 10th Workshop on Multimedia Signal Processing*. 2008, pp. 791–796. Available from DOI: 10.1109/MMSP.2008.4665182.
33. KODOVSKÝ, Jan; SEDIGHI, Vahid; FRIDRICH, Jessica. Study of cover source mismatch in steganalysis and ways to mitigate its impact. In: *Media Watermarking, Security, and Forensics 2014*. International Society for Optics and Photonics, 2014, vol. 9028, 90280J.
34. COGRANNE, Rémi; GIBOULOT, Quentin; BAS, Patrick. The ALASKA steganalysis challenge: A first step towards steganalysis. In: *Proceedings of the ACM Workshop on Information Hiding and Multimedia Security*. 2019, pp. 125–137.
35. COVER, Thomas M. *Elements of information theory*. John Wiley & Sons, 1999.
36. PEVNÝ, Tomáš; ŠEPÁK, Dominik; ADAM, Lukáš. *Algorithmic Approaches to Cover-source Mismatch*. 2022. In review.
37. BEZANSON, Jeff; KARPINSKI, Stefan; SHAH, Viral B; EDELMAN, Alan. Julia: A fast dynamic language for technical computing. *arXiv preprint arXiv:1209.5145*. 2012.
38. BESARD, Tim; FOKET, Christophe; DE SUTTER, Bjorn. Effective Extensible Programming: Unleashing Julia on GPUs. *IEEE Transactions on Parallel and Distributed Systems*. 2018. ISSN 1045-9219. Available from DOI: 10.1109/TPDS.2018.2872064.
39. BESARD, Tim; CHURAVY, Valentin; EDELMAN, Alan; DE SUTTER, Bjorn. Rapid software prototyping for heterogeneous and distributed platforms. *Advances in Engineering Software*. 2019, vol. 132, pp. 29–46.
40. INNES, Michael; SABA, Elliot; FISCHER, Keno; GANDHI, Dhairya; RUDILOSSO, Marco Concetto; JOY, Neethu Mariya; KARMAI, Tejan; PAL, Avik; SHAH, Viral. Fashionable Modelling with Flux. *CoRR*. 2018, vol. abs/1811.01457. Available from arXiv: 1811.01457.
41. INNES, Mike. Flux: Elegant Machine Learning with Julia. *Journal of Open Source Software*. 2018. Available from DOI: 10.21105/joss.00602.
42. DUNNING, Iain; HUCHETTE, Joey; LUBIN, Miles. JuMP: A Modeling Language for Mathematical Optimization. *SIAM Review*. 2017, vol. 59, no. 2, pp. 295–320. Available from DOI: 10.1137/15M1020575.

43. DANISCH, Simon; KRUMBIEGEL, Julius. Makie.jl: Flexible high-performance data visualization for Julia. *Journal of Open Source Software*. 2021, vol. 6, no. 65, p. 3349. Available from DOI: [10.21105/joss.03349](https://doi.org/10.21105/joss.03349).

Appendix A

Full Tables From the Experimental Section

| | | A QF75 | B QF75 | C QF75 | D QF75 | E QF75 | F QF75 | G QF75 | diagonal |
|---|-------|------------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|----------|
| A | QF75 | 7.9 \pm 1.43 | 18.9 \pm 4.43 | 8.1 \pm 1.38 | 46.2 \pm 0.87 | 12.8 \pm 1.26 | 10.2 \pm 0.66 | 48.1 \pm 1.31 | |
| B | QF75 | 19.2 \pm 0.98 | 5.7 \pm 0.72 | 22.1 \pm 3.70 | 47.2 \pm 0.47 | 18.1 \pm 2.39 | 13.8 \pm 0.98 | 49.1 \pm 0.89 | |
| C | QF75 | 10.4 \pm 0.40 | 22.3 \pm 3.58 | 9.0 \pm 1.37 | 48.1 \pm 1.11 | 16.2 \pm 1.20 | 12.7 \pm 0.48 | 48.1 \pm 0.87 | |
| D | QF75 | 10.2 \pm 4.88 | 49.6 \pm 0.23 | 9.3 \pm 2.93 | 2.0 \pm 0.79 | 14.5 \pm 5.85 | 13.3 \pm 7.21 | 41.5 \pm 6.57 | |
| E | QF75 | 17.8 \pm 1.71 | 24.2 \pm 4.43 | 19.7 \pm 0.86 | 48.3 \pm 0.78 | 8.3 \pm 0.47 | 13.4 \pm 2.02 | 49.1 \pm 0.31 | |
| F | QF75 | 7.2 \pm 0.52 | 17.6 \pm 4.70 | 6.6 \pm 0.90 | 46.4 \pm 0.60 | 10.7 \pm 2.49 | 7.2 \pm 1.31 | 47.9 \pm 1.08 | |
| G | QF75 | 26.8 \pm 10.01 | 47.2 \pm 1.24 | 26.0 \pm 5.96 | 23.6 \pm 5.06 | 24.4 \pm 9.30 | 14.7 \pm 2.94 | 11.5 \pm 3.28 | |
| A | QF100 | 49.8 \pm 0.21 | 49.9 \pm 0.12 | 49.9 \pm 0.33 | 50.0 \pm 0.00 | 49.9 \pm 0.25 | 50.0 \pm 0.00 | 50.0 \pm 0.00 | |
| B | QF100 | 48.7 \pm 1.16 | 50.0 \pm 0.00 | 48.6 \pm 1.08 | 50.0 \pm 0.00 | 49.9 \pm 0.14 | 49.4 \pm 1.09 | 49.2 \pm 0.98 | |
| C | QF100 | 49.6 \pm 0.33 | 49.9 \pm 0.18 | 49.6 \pm 0.51 | 50.0 \pm 0.00 | 49.9 \pm 0.24 | 49.9 \pm 0.09 | 50.0 \pm 0.00 | |
| D | QF100 | 49.8 \pm 0.26 | 49.6 \pm 0.61 | 49.8 \pm 0.23 | 50.0 \pm 0.00 | 50.0 \pm 0.00 | 49.8 \pm 0.26 | 50.0 \pm 0.00 | |
| E | QF100 | 49.6 \pm 0.41 | 48.8 \pm 1.09 | 49.8 \pm 0.16 | 50.0 \pm 0.00 | 50.0 \pm 0.00 | 49.8 \pm 0.31 | 50.0 \pm 0.00 | |
| F | QF100 | 49.9 \pm 0.17 | 50.1 \pm 0.17 | 49.6 \pm 0.46 | 50.0 \pm 0.00 | 49.7 \pm 0.00 | 49.9 \pm 0.17 | 50.0 \pm 0.00 | |
| G | QF100 | 50.0 \pm 0.00 | 49.8 \pm 0.34 | 50.0 \pm 0.00 | 50.0 \pm 0.00 | 49.6 \pm 0.69 | 49.8 \pm 0.34 | 50.0 \pm 0.00 | |
| | | A QF100 | B QF100 | C QF100 | D QF100 | E QF100 | F QF100 | G QF100 | |
| A | QF75 | 49.1 \pm 1.61 | 50.0 \pm 0.00 | 49.8 \pm 0.37 | 50.0 \pm 0.00 | 49.9 \pm 0.12 | 49.9 \pm 0.12 | 50.0 \pm 0.00 | 7.9 |
| B | QF75 | 49.4 \pm 0.89 | 49.2 \pm 0.68 | 49.8 \pm 0.62 | 50.0 \pm 0.00 | 49.4 \pm 0.59 | 48.9 \pm 0.14 | 50.0 \pm 0.00 | 5.7 |
| C | QF75 | 49.2 \pm 1.37 | 50.0 \pm 0.00 | 49.7 \pm 0.46 | 50.0 \pm 0.00 | 49.7 \pm 0.46 | 49.8 \pm 0.24 | 50.0 \pm 0.00 | 9.0 |
| D | QF75 | 47.4 \pm 4.42 | 50.0 \pm 0.00 | 49.6 \pm 0.61 | 50.0 \pm 0.00 | 50.0 \pm 0.00 | 50.0 \pm 0.00 | 50.0 \pm 0.00 | 2.0 |
| E | QF75 | 48.9 \pm 2.84 | 49.4 \pm 1.09 | 49.8 \pm 0.31 | 50.0 \pm 0.00 | 49.8 \pm 0.16 | 49.7 \pm 0.00 | 50.0 \pm 0.00 | 8.3 |
| F | QF75 | 49.2 \pm 1.38 | 50.0 \pm 0.00 | 49.9 \pm 0.17 | 50.0 \pm 0.00 | 49.9 \pm 0.17 | 49.9 \pm 0.17 | 50.0 \pm 0.00 | 7.2 |
| G | QF75 | 49.4 \pm 1.03 | 50.0 \pm 0.00 | 49.4 \pm 1.03 | 50.0 \pm 0.00 | 50.0 \pm 0.00 | 49.8 \pm 0.34 | 50.0 \pm 0.00 | 11.5 |
| A | QF100 | 40.7 \pm 0.62 | 49.5 \pm 0.33 | 40.8 \pm 0.81 | 49.4 \pm 0.50 | 41.0 \pm 1.70 | 41.4 \pm 0.57 | 48.6 \pm 0.12 | 40.7 |
| B | QF100 | 47.5 \pm 0.72 | 34.0 \pm 1.42 | 48.7 \pm 1.06 | 49.9 \pm 0.14 | 48.0 \pm 0.76 | 48.5 \pm 1.42 | 49.5 \pm 0.62 | 34.0 |
| C | QF100 | 40.9 \pm 0.37 | 48.8 \pm 0.48 | 41.4 \pm 0.73 | 48.8 \pm 0.51 | 42.8 \pm 1.59 | 41.5 \pm 1.35 | 47.0 \pm 0.24 | 41.4 |
| D | QF100 | 43.1 \pm 1.54 | 46.5 \pm 1.11 | 44.1 \pm 1.20 | 22.4 \pm 1.36 | 43.7 \pm 1.44 | 43.1 \pm 0.97 | 44.3 \pm 0.90 | 22.4 |
| E | QF100 | 39.2 \pm 1.86 | 49.6 \pm 0.41 | 41.3 \pm 1.48 | 48.6 \pm 1.64 | 38.6 \pm 1.48 | 39.5 \pm 2.15 | 47.7 \pm 1.62 | 38.6 |
| F | QF100 | 39.2 \pm 0.30 | 49.2 \pm 0.75 | 41.2 \pm 0.86 | 49.2 \pm 0.62 | 39.0 \pm 1.99 | 39.8 \pm 1.67 | 47.8 \pm 0.62 | 39.8 |
| G | QF100 | 46.0 \pm 1.50 | 49.4 \pm 0.60 | 46.4 \pm 1.79 | 45.0 \pm 0.34 | 45.4 \pm 2.25 | 44.6 \pm 0.00 | 44.4 \pm 2.41 | 44.4 |

Table A.1: Probability of error P_E in percents on the test dataset. Horizontally are cover sources on which the classifiers were trained, vertically cover sources on which the classifiers were tested. The diagonal column is diagonal of the table. Closed form solution of ridge regression, message length of 0.04 bpAC, nsF5 embedding, cc-JRM features.

| | | | diagonal | holistic linear | holistic MLP | minregret simple train | min- regret simple validation | minregret extended validation | minrisk of RP | minregret convex | atomistic QF predicted | atomistic QF known |
|------------|-----|-------|----------|-----------------|-----------------|---------------------------|-------------------------------------|-------------------------------------|-----------------|---------------------|---------------------------|-----------------------|
| | A | QF75 | 7.9 | 0.4 ± 0.87 | -1.6 ± 1.62 | 0.4 ± 0.75 | 1.1 ± 0.89 | 1.5 ± 0.12 | 17.2 ± 2.81 | -0.8 ± 0.98 | -0.1 ± 1.86 | -0.2 ± 1.99 |
| | B | QF75 | 5.7 | -0.1 ± 1.55 | -0.3 ± 0.41 | 1.3 ± 1.08 | 0.6 ± 1.23 | 2.0 ± 0.27 | 24.8 ± 2.71 | 0.3 ± 0.89 | 0.3 ± 0.59 | 0.1 ± 0.76 |
| | C | QF75 | 9.0 | -0.3 ± 0.79 | -0.1 ± 1.28 | 0.8 ± 0.57 | 0.6 ± 0.57 | 0.5 ± 1.32 | 21.2 ± 0.97 | 0.5 ± 0.55 | 0.5 ± 1.65 | 0.5 ± 1.65 |
| | D | QF75 | 2.0 | 0.7 ± 0.63 | 0.4 ± 0.31 | 2.9 ± 2.04 | 1.0 ± 0.77 | 0.6 ± 0.45 | 12.1 ± 7.03 | 0.1 ± 0.48 | -0.0 ± 0.79 | -0.0 ± 0.79 |
| | E | QF75 | 8.3 | -2.1 ± 0.86 | -2.6 ± 1.09 | -1.4 ± 1.83 | -1.3 ± 1.23 | -1.2 ± 1.89 | 14.2 ± 5.75 | -1.9 ± 0.00 | 0.2 ± 1.12 | 0.0 ± 1.23 |
| | F | QF75 | 7.2 | -2.3 ± 0.46 | -1.0 ± 0.91 | -0.4 ± 1.70 | -1.0 ± 0.69 | -0.1 ± 1.05 | 15.7 ± 5.88 | -0.6 ± 1.56 | -0.2 ± 0.86 | -0.2 ± 0.62 |
| | G | QF75 | 11.5 | -1.0 ± 1.82 | 2.2 ± 1.03 | 4.2 ± 1.91 | 3.4 ± 1.79 | -0.0 ± 3.00 | 35.7 ± 2.81 | 0.8 ± 3.05 | 11.1 ± 5.19 | 9.9 ± 5.19 |
| | A | QF100 | 40.7 | -1.3 ± 2.37 | -3.4 ± 0.66 | -2.1 ± 1.64 | -0.3 ± 1.01 | -1.2 ± 0.21 | -1.2 ± 0.37 | -2.7 ± 1.50 | -0.7 ± 0.99 | -0.0 ± 0.45 |
| | B | QF100 | 34.0 | -2.5 ± 2.48 | -0.9 ± 1.80 | 0.2 ± 1.34 | 0.9 ± 1.53 | 0.7 ± 0.85 | 4.2 ± 3.86 | 1.9 ± 0.94 | 0.1 ± 1.30 | 0.0 ± 1.42 |
| | C | QF100 | 41.4 | -2.0 ± 1.44 | -2.5 ± 0.84 | -2.8 ± 2.48 | -1.0 ± 1.27 | -0.8 ± 1.30 | -2.2 ± 0.88 | -1.8 ± 1.32 | 0.2 ± 1.14 | 0.2 ± 0.95 |
| | D | QF100 | 22.4 | 11.3 ± 0.48 | 7.0 ± 1.56 | 1.8 ± 1.21 | -0.3 ± 1.42 | 0.3 ± 1.40 | 16.7 ± 2.58 | -1.6 ± 0.54 | -0.0 ± 1.36 | -0.0 ± 1.36 |
| | E | QF100 | 38.6 | -0.4 ± 1.02 | -1.6 ± 0.62 | -1.5 ± 0.27 | 1.4 ± 2.34 | 0.3 ± 0.82 | -0.3 ± 0.16 | -1.7 ± 2.42 | -0.4 ± 1.62 | -0.2 ± 1.50 |
| | F | QF100 | 39.8 | -0.8 ± 3.30 | -2.6 ± 1.92 | -0.3 ± 0.30 | 1.3 ± 0.91 | -0.4 ± 0.96 | -0.5 ± 3.01 | -2.2 ± 1.41 | 0.3 ± 0.52 | 0.9 ± 0.30 |
| | G | QF100 | 44.4 | -0.6 ± 0.91 | -2.8 ± 3.72 | -2.6 ± 1.82 | -1.2 ± 2.75 | -2.2 ± 0.60 | -0.6 ± 2.09 | -1.8 ± 1.50 | 0.8 ± 0.00 | 0.6 ± 0.34 |
| max regret | A-E | | | 11.3 | 7.0 | 2.9 | 1.4 | 2.0 | 24.8 | 1.9 | 0.5 | 0.5 |
| mean P_E | | | | 21.4 | 20.4 | 21.0 | 21.3 | 21.3 | 31.7 | 20.2 | 21.0 | 21.0 |
| max P_E | | | | 39.5 | 38.9 | 38.7 | 40.4 | 40.6 | 39.5 | 39.6 | 41.6 | 41.6 |
| max regret | A-G | | | 11.3 | 7.0 | 4.2 | 3.4 | 2.0 | 35.7 | 1.9 | 11.1 | 9.9 |
| mean P_E | | | | 22.3 | 21.6 | 22.4 | 22.7 | 22.4 | 33.6 | 21.5 | 23.2 | 23.2 |
| max P_E | | | | 43.8 | 41.7 | 41.9 | 43.3 | 42.3 | 47.2 | 42.7 | 45.2 | 45.0 |

Table A.2: Probability of error P_E in percents on the test dataset. Vertically are cover sources on which the classifiers were tested and aggregated results from the respective columns. The diagonal column is from the table A.1 Other columns are relative to the diagonal column (positive values are larger than the diagonal, negative lower than the diagonal) and their value is called *regret*. The classifiers were trained only on cover sources A to E and the F and G cover sources were used only for test purposes. Message length of 0.04 bpAC, nsF5 embedding, cc-JRM features.

| | | diagonal | holistic linear | holistic MLP | minregret simple train | min- gret simple validation | minregret extended validation | minrisk of RP | minregret convex | atomistic QF predicted |
|-------------|------|----------|-----------------|------------------|---------------------------|-----------------------------------|-------------------------------------|-----------------|---------------------|---------------------------|
| A | QF76 | 9.4 | 1.3 \pm 1.18 | 6.5 \pm 2.05 | 2.2 \pm 1.12 | 5.6 \pm 5.15 | 4.3 \pm 3.25 | 18.6 \pm 4.50 | 6.9 \pm 1.67 | 3.3 \pm 0.77 |
| B | QF76 | 5.7 | 0.7 \pm 1.21 | 2.2 \pm 1.96 | 1.7 \pm 0.98 | 3.3 \pm 2.91 | 3.6 \pm 2.81 | 25.4 \pm 1.31 | 3.3 \pm 2.06 | 5.5 \pm 3.21 |
| C | QF76 | 9.7 | 2.5 \pm 3.44 | 7.6 \pm 1.83 | 3.6 \pm 3.75 | 7.6 \pm 4.14 | 6.2 \pm 3.33 | 20.8 \pm 2.69 | 7.1 \pm 1.47 | 7.5 \pm 1.06 |
| D | QF76 | 2.2 | 10.8 \pm 3.33 | 21.8 \pm 1.07 | 12.0 \pm 3.52 | 15.8 \pm 4.30 | 15.5 \pm 2.21 | 12.3 \pm 4.22 | 21.4 \pm 6.61 | 18.6 \pm 2.05 |
| E | QF76 | 7.7 | 1.2 \pm 0.47 | 3.1 \pm 0.41 | 1.2 \pm 0.47 | 3.8 \pm 3.92 | 3.6 \pm 4.22 | 15.9 \pm 9.09 | 4.5 \pm 1.09 | 3.9 \pm 0.71 |
| F | QF76 | 7.3 | 2.3 \pm 2.59 | 9.3 \pm 5.01 | 3.1 \pm 4.79 | 6.8 \pm 5.32 | 6.5 \pm 4.47 | 16.7 \pm 4.98 | 7.8 \pm 3.23 | 5.7 \pm 2.51 |
| G | QF76 | 11.9 | 13.1 \pm 9.16 | 21.6 \pm 12.23 | 3.6 \pm 3.62 | 22.6 \pm 5.19 | 9.3 \pm 8.04 | 28.4 \pm 9.82 | 19.8 \pm 15.66 | 28.0 \pm 4.29 |
| max regret | QF76 | | 13.1 | 21.8 | 12.0 | 22.6 | 15.5 | 28.4 | 21.4 | 28.0 |
| mean regret | QF76 | | 4.5 | 10.3 | 3.9 | 9.3 | 7.0 | 19.7 | 10.1 | 10.3 |
| A | QF77 | 9.2 | 1.3 \pm 0.86 | -1.1 \pm 2.03 | 1.6 \pm 1.40 | 2.1 \pm 0.94 | 3.0 \pm 1.41 | 17.5 \pm 4.66 | -0.8 \pm 1.18 | 4.1 \pm 1.87 |
| B | QF77 | 7.8 | 0.2 \pm 1.43 | 0.8 \pm 1.57 | 2.5 \pm 0.83 | 0.8 \pm 0.27 | 2.2 \pm 2.14 | 23.3 \pm 4.03 | -0.3 \pm 1.30 | 1.7 \pm 0.62 |
| C | QF77 | 10.4 | 0.0 \pm 1.29 | 0.4 \pm 1.88 | 2.6 \pm 1.81 | 1.2 \pm 1.10 | 2.3 \pm 2.13 | 20.4 \pm 3.26 | 0.8 \pm 2.82 | 2.5 \pm 1.42 |
| D | QF77 | 2.9 | 4.1 \pm 0.85 | 4.4 \pm 1.75 | 9.4 \pm 0.83 | 7.0 \pm 1.95 | 6.7 \pm 2.58 | 41.8 \pm 4.88 | 4.0 \pm 1.38 | 7.3 \pm 1.13 |
| E | QF77 | 8.6 | 0.1 \pm 1.21 | -1.4 \pm 1.98 | 1.1 \pm 0.81 | 0.9 \pm 1.79 | 1.6 \pm 1.40 | 13.3 \pm 4.04 | 0.4 \pm 3.34 | 2.1 \pm 0.68 |
| F | QF77 | 8.8 | -1.0 \pm 1.04 | -2.3 \pm 2.49 | 0.7 \pm 1.70 | -0.2 \pm 1.99 | -0.5 \pm 2.12 | 16.5 \pm 7.82 | -1.2 \pm 1.99 | 1.3 \pm 1.35 |
| G | QF77 | 13.7 | 1.0 \pm 2.81 | 15.1 \pm 12.72 | 5.4 \pm 1.19 | 11.7 \pm 3.38 | 0.4 \pm 2.68 | 36.3 \pm 1.19 | 11.9 \pm 11.36 | 23.8 \pm 3.57 |
| max regret | QF77 | | 4.1 | 15.1 | 9.4 | 11.7 | 6.7 | 41.8 | 11.9 | 23.8 |
| mean regret | QF77 | | 0.8 | 2.3 | 3.3 | 3.4 | 2.2 | 24.1 | 2.1 | 6.1 |
| A | QF90 | 15.5 | 5.4 \pm 3.04 | 11.4 \pm 4.95 | 6.3 \pm 2.16 | 7.4 \pm 1.99 | 5.7 \pm 4.51 | 29.9 \pm 2.12 | 6.8 \pm 2.73 | 20.8 \pm 0.77 |
| B | QF90 | 13.1 | 13.2 \pm 1.36 | 17.9 \pm 1.44 | 13.6 \pm 1.43 | 16.3 \pm 2.23 | 16.8 \pm 3.21 | 24.8 \pm 3.14 | 14.3 \pm 1.08 | 26.7 \pm 2.06 |
| C | QF90 | 19.3 | 3.0 \pm 4.52 | 9.4 \pm 4.47 | 6.0 \pm 1.26 | 7.0 \pm 0.73 | 5.6 \pm 3.15 | 25.4 \pm 2.11 | 7.2 \pm 2.55 | 18.0 \pm 1.81 |
| D | QF90 | 8.7 | 14.5 \pm 2.30 | 30.0 \pm 16.17 | 19.6 \pm 8.28 | 22.8 \pm 10.03 | 20.2 \pm 7.09 | 35.2 \pm 9.67 | 20.9 \pm 6.23 | 39.4 \pm 3.04 |
| E | QF90 | 16.5 | 8.4 \pm 4.53 | 3.5 \pm 1.73 | 2.7 \pm 3.61 | 2.7 \pm 3.34 | 1.8 \pm 1.42 | 25.4 \pm 6.72 | 3.7 \pm 4.14 | 11.4 \pm 3.77 |
| F | QF90 | 17.5 | 5.8 \pm 7.82 | 7.0 \pm 6.15 | 3.3 \pm 3.31 | 4.9 \pm 3.40 | 3.2 \pm 1.67 | 26.2 \pm 4.32 | 3.9 \pm 5.01 | 18.2 \pm 0.52 |
| G | QF90 | 24.2 | 4.6 \pm 1.82 | 19.2 \pm 6.21 | 12.3 \pm 5.84 | 10.5 \pm 1.37 | 11.7 \pm 8.63 | 22.2 \pm 6.19 | 21.8 \pm 3.49 | 19.2 \pm 6.63 |
| max regret | QF90 | | 14.5 | 30.0 | 19.6 | 22.8 | 20.2 | 35.2 | 21.8 | 39.4 |
| mean regret | QF90 | | 7.8 | 14.1 | 9.1 | 10.2 | 9.3 | 27.0 | 11.2 | 21.9 |

Table A.3: P_E in percent on the test dataset of cover sources with quality factors 76, 75, and 90. Vertically are cover sources on which the classifiers were tested and column aggregated results from the respective parts of the table. The classifiers in the columns are the same as in the table A.2 and cover sources with quality factors in this table were **not** exposed to the classifiers during the training. Values to the right of the diagonal column are relative to it (positive values are larger than the diagonal, negative lower than the diagonal). The diagonal column was constructed analogously to the diagonal column in table A.1 – for each row, a single cover source classifier was trained on the train dataset of the cover source matching the row description and evaluated on test dataset of the same cover source, using the closed-form solution of Ridge regression. Message length of 0.04 bpAC, nsF5 embedding, cc-JRM features.