Support Vector Machines
EECE 580B

Lecture 25

April 29, 2010

Jessica Fridrich, Jan Kodovský

Steganalysis

Objective

- **Traditional steganalysis**: a steganography system is considered broken, when the mere *presence* of a hidden message is detected.
- **Forensic steganalysis**: detection of the message may not be sufficient; often, other information would be useful
  - determine type of embedding algorithm (LSB, SS)
  - identify stego software used (F5, OutGuess, Steganos, ...)
  - search for stego key if necessary
  - extract hidden bitstream
  - decrypt the message (cryptanalysis)

Steganalysis

- Steganalysis is a binary hypothesis testing problem – H_0 : cover and H_1 : stego.
- If the distributions were known, Likelihood Ratio Test (LRT) would be the *optimal* detector.
- In absence of distributions, we resort to classification using machine learning.
- Output of steganalysis can be a real number, rather than binary {cover,stego}. If this number is an estimate of the message length (change rate), we speak of *quantitative steganalysis* (SVR useful here!).

Feature-based steganalysis

- Steganalyzer can be built to detect a specific stegosystem (targeted steganalyzer) or to detect an arbitrary stegosystem (universal "blind").
- Constructing a steganalyzer involves the following steps:
  - select good features (sensitive to embedding, insensitive to image content, low dimensionality).
  - if features low-dimensional (e.g., 1D), estimate distributions, use LRT.
  - if features high-dimensional, select a machine-learning tool, e.g., SVM.
  - train SVM on a large and diverse database of cover and stego images, use a mixture of payloads.
- Caveat
  - steganalyzer will depend on the chosen features and the database (scans of photographs, decompressed JPEGs, raw never-compressed digital camera images, processed (denoised) images).

SPAM features (motivation)

Observation

- Neighboring pixels in natural images exhibit dependencies.
- Pixel noise is not iid but also dependent (and dependent on content) due to in-camera image processing, compression, etc.
- *Stego noise in LSB embedding or SSS is pixel-to-pixel independent* and often *idenpendent of content*.

Idea

Model dependencies between neighboring pixels and detect violations of the model due to stego noise.

Modeling dependencies between pixels (1)

Histogram of pixel pairs

The counts of neighboring pixel-pairs, triples, quadruples, ..., will capture all dependencies.

Disadvantages:

- The number of bins grows exponentially with bin size.
- Estimates of some bins may be very noisy.
- High influence of image content.

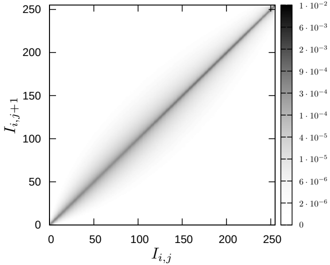


Fig: Probability of co-occurrence

Modeling dependencies between pixels (2)

Differences of pixel values

We model *differences* of pixels instead of the pixel values themselves.

Advantages:

- Image content is suppressed.
- Differences $I_{i,j+1} - I_{i,j}$ are almost independent of $I_{i,j}$.
- Simplification of the model.
- Can be modeled by Markov chains.

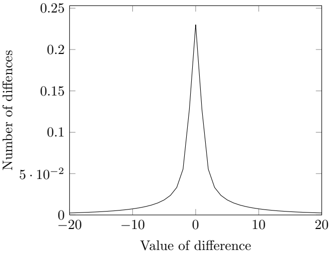
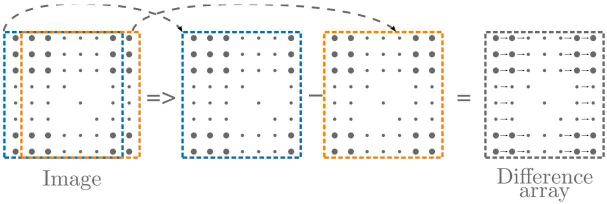


Fig: Histogram of differences

SPAM features (1)

Computing transition probabilities in the horizontal direction:



1. Calculate the difference array, $\mathbf{D}^{\rightarrow}_{i,j} = I_{i,j} - I_{i,j+1}$.
2. Truncate: if $|\mathbf{D}^{\rightarrow}_{i,j}| > T$, set $\mathbf{D}^{\rightarrow}_{i,j} = sign(\mathbf{D}^{\rightarrow}_{i,j})T$.
3. Compute the transition probability matrix

$$\mathbf{M}^{\rightarrow}_{u,v} = Pr(\mathbf{D}^{\rightarrow}_{i,j+1} = u | \mathbf{D}^{\rightarrow}_{i,j} = v), u, v \in \{-T, \ldots, T\}$$

$$\mathbf{M}^{\rightarrow}_{u,v,w} = Pr(\mathbf{D}^{\rightarrow}_{i,j+2} = u | \mathbf{D}^{\rightarrow}_{i,j+1} = v, \mathbf{D}^{\rightarrow}_{i,j} = w,), u, v, w \in \{-T, \ldots, T\}$$

SPAM features (2)

1. Transition matrices $\mathbf{M}^{\cdot}_{\cdot}$ are calculated along 8 directions $\leftarrow, \rightarrow, \downarrow, \uparrow, \nwarrow, \searrow, \swarrow, \nearrow$
2. Features $\mathbf{F}$ are formed from $\mathbf{M}$ by averaging to reduce dimensionality

$$
\begin{aligned}
\mathbf{F}^{\cdot}_{1,\dots,k} &= \frac{1}{4}\left[\mathbf{M}^{\rightarrow}_{\cdot} + \mathbf{M}^{\leftarrow}_{\cdot} + \mathbf{M}^{\downarrow}_{\cdot} + \mathbf{M}^{\uparrow}_{\cdot}\right], \\
\mathbf{F}^{\cdot}_{k+1,\dots,2k} &= \frac{1}{4}\left[\mathbf{M}^{\searrow}_{\cdot} + \mathbf{M}^{\nwarrow}_{\cdot} + \mathbf{M}^{\swarrow}_{\cdot} + \mathbf{M}^{\nearrow}_{\cdot}\right].
\end{aligned}
$$

3. The total number of features is $dim = 2(2T+1)^2$.
4. In your final project, $T = 4 \Rightarrow dim = 2 \times 9^2 = 162$.

Comparison to prior art

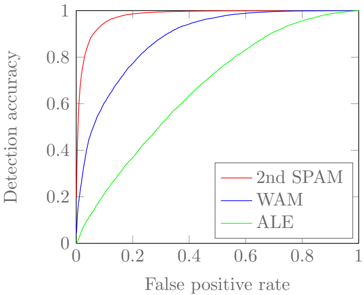


Fig: Payload 0.25 bits per pixel

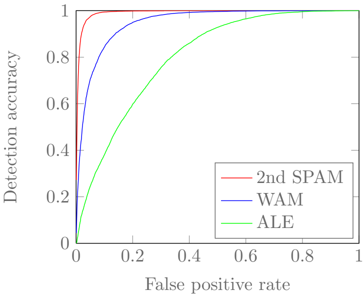


Fig: Payload 0.5 bits per pixel.